'68' MICRO JOURNAL

$2.95USA

Australia	A $ 4.75	New Zealand	NZ $ 4.75
Singapore	S $ 9.45	Hong Kong	H $23.50
Malaysia	M $ 9.45	Sweden	30:-SEK

VOLUME V ISSUE I • Devoted to the 68XX User • January 1983

"Small Computers Doing Big Things"

YOUR CHOICE-smart either way

- Over 140 software driven functions
- 82 x 24 or 82 x 20 screen format — software selectable
- High resolution 7 x 12 matrix characters — P-31 green phosphor
- Upper/lower case character set — plus graphics character set
- 56-key alphanumeric keyboard — plus 12-key cursor, numeric pad
- Internal editing functions — insert, delete, scroll, roll, slide, etc.
- Parallel printer I/O port
- 50 to 38,400 baud operation — programmable
- Cursor type, cursor position, print control characters, protected fields, shift inversion, dual intensity and many other features

8212 — twelve-inch diagnonal screen or **8209** — nine-inch diagnonal screen

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

Pascal for 6809

Pascal for the 6809 is a true native code compiler. Unlike the usual P-code Pascals which run in an interpretive manner, ours produces efficient assembly language mnemonics which can be assembled and run directly. This compiler is available for both 6809 FLEX™ and UniFLEX™. Many features not found in other Pascal systems were implemented while avoiding those features completely non-standard. Features of the Pascal system include:

- Supports most of Jensen and Wirth specification
- Produces fast and efficient 6809, native code
- FLEX run-time package may be trimmed
- Double precision real numbers (16.8 digits)
- Implements scalar, subrange and structured data types
- Standard I/0 using file buffer pointers
- Dynamic storage allocation
- Ability to call other Pascal programs
- FLEX version may call assembly language programs
- Buffered or single character terminal input
- Standard math functions: SIN, COS, ARCTAN, EXP, LN, SQR, SQRT
- Random number generator function
- Many usable, sample programs included
- UniFLEX version supports:
  - Random file positioning
  - Ability to call various UniFLEX system routines
  - Ability to execute UniFLEX utility commands

Pascal on diskette for 5" and 8" 6809 FLEX is available for $200.00 The 5" version requires two disk drives. The UniFLEX version is $300.00 and includes one year of maintenance. All orders should include 3 percent for postage and handling (10 percent on foreign orders).

™FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc.

technical systems consultants, inc.

111 Providence Rd., Chapel Hill, N.C. 27514

(919) 493-1451

'68' MICRO JOURNAL

Portions of text prepared using the following.

SWTPC 6800-6809-OMAF2-CDS1-CT82-Sprint 3
Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216

EDITOR - WORD PROCESSOR
Technical Systems Consultants, Inc.
Box 2573, W. Lafayette, IN 47906
FLEX is TM of TSC

GIMIX Super Mainframe-Assorted memory boards
GIMIX Inc.
1337 West 37th Place
Chicago, Il 60609

CONTENTS

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
615 842-4600

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tenn. and additional entries. Postmaster: send Form 3579 to 68' Micro Journal, PO Box 849, Hixson, Tennessee.

SUBSCRIPTION RATES

USA

1-Year $24.50 2-Years $42.50 3-Years $64.50

FOREIGN

See Page 52

Items Submitted for Publication

Articles submitted for publication should be accompanied by the authors **full name, address, date and telephone number.** It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will be returned.

The following TSC Text Processor commands **ONLY** should be used (due to our proportional processor): .sp space, .pp paragraph, .fi fill and .nf no fill. Also please do not format within the text with multiple spaces. The rest we will enter at time of editing.

STYLO commands are all acceptable except the .pg page command, we print edited text files in continous text.

All articles submitted on diskettes should be in TSC FLEX™ format, either FLEX2 6800, or FLEX9 6809 any version.

If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submitted articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single spaced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). **Please use a dark ribbon!**

All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well as 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good taste'.

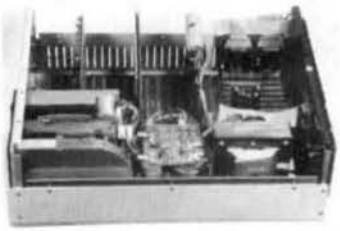
Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less $7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

FLEX - OS-9 LEVEL ONE - UNIFLEX - OS-9 LEVEL TWO

ONLY GIMIX Systems can be configured to run any of these.

GIMIX systems utilize the most powerful 6809 operating systems: FLEX, UniFLEX, OS-9 LEVEL ONE and TWO -- the systems the PROs use. This means a wide selection of software to choose from as well the ability to develop sophisticated, multi-user/multi-tasking programs on your **GIMIX** System.



The GIMIX CLASSY CHASSIS™ consists of a heavy-weight aluminum mainframe cabinet which provides more than ample protection for the electronics and t or 2 optional 5¼" drives.

Backpanel connectors can be added for convenient connection of terminals, printers, drives and other peripherals.

A 3 position locking keyswitch enables users to disable the front panel reset button to prevent accidental or unauthorized tampering with the system.

The **GIMIX** system mother board provides fifteen 50 pin slots and eight 30 pin I/O slots -- the most room for expansion of any SS50 system available. The on board baud rate generator features 11 standard baud rates, 75 to 38.4K, for maximum versatility and compatibility with other systems. Extended address decoding allows the I/O block to be addressed anywhere in the t megabyte address space. All components feature Gold plated connectors for a lifetime of solid connections. All boards are fully buffered for maximum system expansion.

Each **GIMIX** Mainframe System is equipped with an industrial quality power supply featuring a **ferro-resonant constant voltage transformer** to insure against problems caused by adverse power input conditions such as A.C. line voltage fluctuations etc. The supply provides 8 volts at 30 amps and plus or minus 16 volts at 5 amps, more than enough capacity to power a fully loaded system and two internal drives.

The 2MHz GIMIX 6809 PLUS CPU board includes a timeof day clock with battery back-up and 6840 programmable timer to provide the programmer with convenient, accurate time reference. Later addition of 9511 or 9512 arithmetic processors is provided for on the board. The unique **GIMIX** design enables software selection of either OS-9 or FLEX, both included in many complete **GIMIX** systems.

GIMIX STATIC RAM boards require no complicated refresh timing cycles or clocks for data retention. GIMIX memory boards are guaranteed for 2 MHz operation with no wait state or clock stretching required.

Our low power NMOS RAM requires less than 3/4 amp at 8V for a fully populated 64K board. For critical situations, our non-volatile 64K byte CMOS static RAM boards with built in battery back-up retain data even with system power removed. A fully charged battery will power this board for a minimum of 21 days. A write protect switch permits CMOS boards to be used for PROM/ROM emulation and software debugging.

The GIMIX DMA controller leaves the processor free to perform other tasks during disk transfers - an important feature for multi-user/multi-tasking systems where processor time allocation is critical. The DMA board will accomodate up to 4 drives 5¼" or 8" in any combination running single or double density single or double headed. Programmed I/O Disk Controllers are also available.

GIMIX systems are designed with ultimate **RELIABILITY** in mind. You can choose from the below featured systems or select from our wide variety of components to build a custom package to suit your needs.

GIMIX 2MHz 6809 System including: CLASSY CHASSIS, 6809 PLUS CPU BOARD, 56KB STATIC RAM, 2 SERIAL PORTS W/CABLES, GMXBUG MONITOR, FLEX, and OS-9 LEVEL 1 ..... $3248.49
FOR TWO 5¼" 40 TRACK OSDD DRIVES ADD ..... $ 900.00
GIMIX 128KB WINCHESTER SYSTEM including: CLASSY CHASSIS, 6809 PLUS CPU BOARD, 128KB STATIC RAM, 4 SERIAL PORTS W/CABLES, 5¼" 80 TRACK DSDD FLOPPY DISK DRIVE, 19MB 5¼" WINCHESTER HARD DISK, OS9 LEVEL 2, EDITOR AND ASSEMBLER $8998.09

50HZ Versions Available. 8" Drives Available — Contact GIMIX for Prices and Information.

The Sun Never Sets On A GIMIX!

GIMIX users are found on every continent, including Antarctica. A representative group of GIMIX users includes: **Government Research and Scientific Organizations** in Australia, Canada, U.K. and in the U.S.; NASA, Oak Ridge, White Plains, Fermilab, Argonne, Scripps, Sloan Kettering, Los Alamos National Labs, AURA. **Universities:** Carleton, Waterloo, Royal Military College, in Canada; Trier in Germany; and in the U.S.; Stanford, SUNY, Harvard, UCSD, Mississippi, Georgia Tech. **Industrial users** in Hong Kong, Malaysia, South Africa, Germany, Sweden, and in the U.S.; GTE, Becton Dickinson, American Hoechst, Monsanto, Allied, Honeywell, Perkin Elmer, Johnson Controls, Associated Press, Aydin, Newkirk Electric, Revere Sugar, HI-G/AMS Controls, Chevron. **Computer mainframe and peripheral manufacturers,** IBM, OKI, Computer Peripherals Inc., Qume, Floating Point Systems. **Software houses;** Microware, T.S.C., Lucidata, Norpak, Talbot, Stylo Systems, AAA, HHH, Frank Hogg Labs, Epstein Associates, Softwest, Dynasolt, Research Resources U.K., Microworks, Mela Lab, Computerized Business Systems.

1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510
TWX 910-221-4055

GIMIX INC.

The Company that delivers

Quality Electronic products since 1975.

Flex User Notes

Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

And So Forth

My column containing the language discussions has just been published (September issue), and my prediction of communication from FORTH users has been fulfilled. I received a call all the way from California from a FORTH fan who thinks I'm crazy to say that FORTH programs are hard to read. He said that hard to read programs were generally written by programmers who had been used to BASIC and Assembler, and who use cryptic names for variables and procedures (word definitions in FORTH). I've asked him to send me some examples of FORTH programs that are readable. If I have any further thoughts on FORTH, after looking at a readable program, I will present them here.

I repeat what I tried to say, apparently unsuccessfully, in the language discussion. I never said FORTH is not a good language. I never advised any of you not to use it. I simply said that I had spent considerable time learning it, and that for me, it is harder to use than Assembler. How about one of you FORTH fans out there writing an article for '68' to tell us why you think FORTH is so great?

You (FORTH fans) tell me that there is no floating point math in FORTH because each application demands different precision and I might as well write one for myself to the precision I need. Fine, but I've already written a couple of floating point packages, one for the 6800, and another for the 6809. My main reason for going to a complied language is to avoid having to do that again! Maybe we're not communicating because we are not thinking of the same sorts of applications. In the things that I use for my work, a 2000 to 3000 line Pascal program is not unusual. That would translate to something on the order of 40 to 60 FORTH screens. That is without having to write my own floating point package. Somehow I don't see three people working together on a FORTH program, as we have had to do in our large application program in Pascal.

I recently wrote a set of programs (in Pascal) to do mailing list functions. The set includes a program to create and add to a mailing list data file, one to change records in the file, one to delete records in the file, and one to sort records from one sequential file to another, (by name or zip code).

The records contain name, (including Mr. & Mrs. etc.) Address, Optional extra address line, City, State, Zip, and Telephone Number. Also included is a Print Label program that prints standard 3.5 by 1 inch labels with all the information but the phone number, and a Print Directory (prints pages for a personal directory) program. In Pascal, the whole thing took about three evenings, though I admit I had seen a similar set in BASIC some time before. I did not, however, use the BASIC program as a reference for the project.

Someone send me a set in FORTH that works either in Talbot Microsystems FORTH (very nearly fig FORTH) or XFORTH, the FLEX compatible version, and tell me that it took less than 15 hours to write and debug. If it is reasonably readable, you might make a believer out of me. If I am correct, Fig FORTH doesn't have the capability of accessing data files...unless of course, you write your own.

Diversity of Software

Recently I've noted a great supply of new software for our 6809 systems. In the past few months, several assemblers have appeared. First I reported CRASMB, the cross assembler that runs on the 6809 (FLEX) and will cross asssemble code for all of the common microprocessors. I recently received a copy of MACE, a 6809 assembler by Windrush. They followed that with a copy of a 6800 cross assembler. I have yet another on the way for a look. I hardly need mention all the versions of "C" that have been appearing on the market. I'm pleased to see such a choice.

Along that line, I need to take some time and space here to relate some observations about "that other processor" and "that other operating system". A friend asked me for advice on a business system. Having heard all the stories about how much software is available for CP/M (and/or MP/M), I indicated with some reluctance that he should go get a system that is compatible with that software.

Well, the system arrived, and I'm rather completely turned off about it. The best thing about OUR systems is the completeness of information that we have. TSC saw fit to provide the Advanced Programmer's Guide so we could interface our programs and our hardware with the operating system very efficiently. I have written all sorts of print routines, for example for my system, to run several different printers. That other system is just not so nice. Want to change printers? Add a modem? The procedure is to play 100 questions with a utility called GENSYS. If you answer all of them correctly, you have reconfigured the system. Plug the modem into the proper RS-232 connector, and it all works. Answer one question wrong, and disaster strikes. Want to write your own drivers? Impossible. No information available on the hardware or the system software!

Perhaps the very reason that I like our systems is a valid reason for some other folks to like the other system. You don't have to know anything about the hardware and/or operating system to reconfigure the system. While the user of such a system can certainly become a proficient computer user, we can surely surpass him with our knowledge of an operating system, and the hardware in our systems. In my case, because I design computer systems for end use in instruments and machine controls, the later sort of system is vastly more useful to me.

But what about all that good software? Baloney! I realize that I can be comfortable here in familiar surroundings with all you 68XX fans reading this, but honestly, I've tried WORDSTAR, one of the best the other guys have to offer. STYLOGRAPH will do everything WORDSTAR can do, and to it in a much more direct and simple manner, and furthermore do it in much less time. If you are familiar with STYLOGRAPH, you know that you may have a very large text file, set up, for example, for a line length of, say 80. You may change the entire file to a line length of 64 by changing one line of the file from ll 80 to ll 64. With WORDSTAR, you must go through the entire file, positioning the cursor at the start of each paragraph, and typing^B (after going through a sequence of keystrokes to set the right margin to a different value).

I have little doubt that Bob Bundy, who wrote STYLOGRAPH, was very familiar with WORDSTAR, and patterned STYLOGRAPH closely after it, but in the process, he made it much easier to use! Of course we all like something familiar over something new and mysterious, but I think I am being more objective than that.

AHA!! My feelings have been strongly reinforced. My friend has a very capable secretary who has used word processors before. About three years ago we had a 6800 system set up with TSC EDIT and PR. She is deeply involved in using the new system for very similar purposes. This morning she said "You know, I really don't see that this new system is all that much better than

that old one I used back at ....". I smiled a bit as she went on to say that Wordstar could of course do everything the old system could do, and show the finished format right on the screen, but its operation is so complex and there are so many things to remember, that it takes a long time to get anything done. We both agreed that perhaps some of the trouble was our lack of familiarity with both Wordstar and the Z80 system that we have.

This afternoon, she asked if we couldn't set up some of the special function keys on the Televideo terminal to take the place of some of the three or four control character sequences always needed in Wordstar. Sigh! If that were a 6809 system, I would write a TERMINIT.CMD that would send all the proper sequences of characters to the terminal to initialize those function keys. With this MP/M system, I could probably figure it out in a few days, but I don't have that much time to spend right now. I'm beginning to be sorry that I didn't just go with a recommendation of a 6809 system.

I just received my October '68' Micro Journal, and read with interest the comments of Don Williams under the heading "Sorry, I don't believe it". Reading that has prompted me to add a bit more here to reinforce Don's comments. I have one of those VERY early SWTPC 6800 boxes. I bought it as a kit over four and a half years ago. It has the original MP-M mother board in it. It came with the processor board (MP-A), a pseudo serial interface (MP-C), and a 4K memory board with 2K worth of chips installed. Delivery was about 4 months from date of order. The original 8K BASIC cost (if MY memory is not completely shot) about $15.

Don's point was that some of those old systems are still running today. That is absolutely true in my case. First addition was a MF-68 box with two 5" single sided single density 35 track disk drives. Wow! How could I ever use up enough disk space so that two drives with 96K or so each wouldn't be enough? I kept adding memory until I had 56K. At that point, my power supply was a bit marginal, so I added another transformer and a heftier bridge for the 8 volt unregulated supply. With each upgrade, there were modifications, generally on the mother board, and consisting of little more than cutting a trace here and there, and adding a wire.

I figured out how, and added a switch to change the I/O decoding back and forth from $8000 to $E000 so I could plug in my old MP-A card or my newer MP-09. I added a pair of 8" drives and a DMAF card. After a few more modifications, I had those drives running with the 6800 or the 6809 also. I went from an IDS roll paper friction feed printer to a Paper Tiger, successfully ran a Heath H14 for a while, borrowed and ran a Centronics 737 to print my book manuscript, and currently have an EPSON MX-80 FT (which I like best of all for the character quality). The only operational feature I lack currently, is the ability to run with a 2 MHZ clock. I think the old mother board is up to it if I would spend the $$ to get a 2 MHZ processor board and replace some slow memory and I/O chips with their 2 MHZ counterparts.

Yes, Don, I agree fully, we really did get a lot for our money. Problems? I've lost a power supply bridge a couple times, until I latched onto one with a bit higher current capacity. A 1488 and or 1489 went out a couple times (RS-232 level converters), and I had one really serious failure of a bus driver, which unfortunataly hung up in the "on" condition and took several other bus drivers and a few other chips with it. I was able to diagnose the problem with a scope and some part swapping, and I had the thing limping in a couple of hours, and, thanks to very quick response from SWTPC on a spare parts order, back up to full operating capabilities again within a week.

With regard to software, we don't have as big a selection as the other guys, but generally, the quality is there, and the suppliers are willing to help when we get into difficulties.

Amazing

Some few months ago, I purchased a TRS Color Computer so I could get in on the advances in computing at a low cost, and have a small system for my family to use for games on the Color TV. Well, I of course received both Data Comp's version of FLEX, and Frank Hogg's. I recently received an update of the Data Comp version, but that is a bit beside the point at the moment. I bought a 64K Color Computer, and a pair of Double Sided Double Density Tandon disk drives. First I found that I had 1404 sectors on my disks that were formatted double sided and double density. (Both Data Comp and Hogg FLEX allow this combination.)

As if that weren't enough, I had the chance this week to borrow an MPI drive that has double track density as well. Because the number of user tracks (track 0 is taken by the DOS) goes from 39 to 79, the number of sectors more than doubles, to 2844! That is over 716K of user bytes on each disk! I plugged the drive in and formatted a disk, then copied my whole system disk onto it with no problem whatever.

Both suppliers have worked out software by means of which you may connect a terminal to the CC, via the serial port. I opened the CC after about 5 seconds thought about voiding the warranty, and connected a standard DB-25 connector to that port, so I could simply plug my terminal cable into it. Having 80 column capability again, I managed to get a screen editor up on the CC. It works flawlessly.

My next project is going to be to design a serial to parallel converter for the serial port so I can run my EPSON printer on the CC. At that point, I will have a reasonably portable word processor system with powerful capabilities. The converter should cost only $20 or so for parts. I will publish the design here if and when it proves to be reliable. While it is not my intention to switch this column to being a CC column, I will include some information on it from time to time.

Music

I've reviewed the Palm Beach Software music board recently. I'd like to keep you informed on continuing development of that project. First of all, the software has all been declared to be Public Domain by Dan Farnsworth of Palm Beach. His music board is hardly going to make him wealthy at the price he is charging. Dan tells me that the main reason for his getting into the hardware project was to get more people interested in Computer Music, so that the library of music could grow faster.

After I played the selections a few times, it occurred to me that there was no way to control the loudness of the music, so I got in and added a way to do it. My first attempt was a rather clumsy patch to Dan's MOZART, the music compiler. Dan has figured out a much better way to include a loudness specification in the music, and has incorporated the changes in the system. They are perfectly compatible with older music files.

I also have an idea for some hardware for you to homebrew a music board, and I will publish that after I get it working successfully. If you are at all musically inclined, you may think as I did, that all computer music is too mechanical and dull. Don't underestimate what can be done with our SS-50 bus systems. Dan's music plays 4 parts very well, and I have been thinking of doing a 5 part version so I can include "modern or extended" chords. If you decide to try the music board, I promise you will be pleasantly surprised. If you already have one, make all of us happy by entering some music on your computer, to be added to the library of selections.

Answer

For those of you who spent time trying to crack my Cryptogram in the October column, I have to say that if you didn't see the solution you overlooked the obvious. The words are just plain nonsense with no meaning at all. The important thing is the length of the words. Read the thing as morse code and you will be successful. Eg. 'nonsense as nonsense is' translates to . . (C) etc. Any word of one or two letters is a "dot" and longer words are "dash". The idea could be extended by figuring out how to code "1" and "0" in terms of word lengths, and representing the ASCII code for letters. Hope you had fun. Of course by now you have figured out that the message is simply "Can you read this".

Feedback

I received a call from Richard Don of Gimix racently, regarding my comments in the October issue about having to pay a premium for 2 MHZ boards because the 1 MHZ are no longer available. I was speaking as a user of SS-50 bus components in an industrial application, not as a hobbyist.

I'm not quite sure I know how to put this, but this is roughly how I understood Richard's point. The price increases (or should I say, today's prices relative to earlier ones) reflect the industry growing up. When SS-50 sales were limited to hobbyists and we bought by mail direct from SWTPC in Texas, the prices reflected the fact that we wouldn't complain if a chip or two failed. In fact, as hobbyists, we would consider it a challenge to find and replace such a chip.

Today, many more SS-50 bus systems are sold to small businesses as business computers and engineering development systems, than are sold to hobbyists. Businesses must depend on their computers. They must be fully tested to insure their reliability, and service must be available locally. That later means that there must be dealers around to support the systems that are sold. Dealers need to have a reasonable markup on the equipment they sell, to keep them in business. The service they provide generally makes the difference between selling a computer system and not selling it.

Richard mentioned that his systems are "burned in" (left with power on running a test program) for two weeks. He said that of a batch of 1000 memory chips, there would be about 6 that are bad initially, and that about 6 more would fail within the two week period, most failures occurring around the 8th or 9th day (no one knows quite why). Since the new larger memory boards hold so many chips, the failure of 6 out of 1000 would represent failures of a very large percentage of the memory boards had the bad chips not been weeded out by the burn in procedure.

From personal experience, I can support Richard's point. I recently spoke to two owners of small businesses, and recommended 6809 systems with Uniflex and a hard disk. (In fact, I recommended Gimix systems). Both settled on NEC computers costing 60% more, and software priced out of sight compared to the 6809 software. Why? Both asked "Who is going to service the system for us?" Richard, had you a dealer in the Detroit area with a store, a means of financing the purchase, and a service department, I think both would have considered your systems a good buy.

A little later in the column I wrote another bit, this time as a hobbyist. I mentioned the TRS Color Computer, and discussed how inexpensively a reasonable system might be put together. I repeat that in that portion of the column I was speaking as a hobbyist. Of course that little CC doesn't have all the capabilities of the larger SS-50 systems. It is not as reliable. It is not as convenient to connect accessories to it. The screen is limited. It's ultimate expansion capacity is not as large.

I once had a friend who would sometimes express a great desire to have some sort of item (for example, a camera). He would then (in his own mind) work his way up from an inexpensive model to the very best. The next step would be to decide that he couldn't afford the best. The result was that he would not buy any camera.

What I was trying to say is that the CC has brought the cost of a "reasonable" computer into a range where many more people can justify it as a hobby investment. You can get started and see if you find the computer interesting, without investing an arm and a leg. There is enough expansion capability there so that you can put together a capable system without "starting over again". If you can justify a greater expense either because the computer will be used for business or educational purposes, or if you happen to be independently wealthy, it would make sense to purchase a more expandable system in the first place.

I'm sorry I switched hats without clearly identifying my "present status" in my discussions in the October column. Richard, if I haven't fully clarified, or understood your point, please get back to me again.

COPY UTILITY

Last but not least, I have included here a listing of a disk copy utility. Another copy utility??? Yes, one that copies a FLEX formatted disk sector for sector. It copies a track at a time from source to destination. Since such utilities copy into corresponding sector addresses with no regard to contents, it only makes sense to allow copying if both disks are formatted with the same number of tracks, and the same number of sectors per track. A disk could be copied to a disk with more tracks and or sectors, but since the System Information Record and the directory are copied exactly, any extra sectors would be lost. I therefore elected to allow copy only if the disks have system information records that agree.

SCOPY copied my 5" 6809 system disk to a freshly formatted disk in about 5 minutes. The standard copy utility took over 15 minutes because there were a large number of small files. Copy must access each disk twice for each file to be copied, once for the directory entry and one for the file proper. SCOPY reads all the sectors on a track into a buffer, and then writes all the sectors to the destination disk. Disk accesses for both disks are equal to the number of tracks on the disk.

There are available some "MIRROR" utilities that exactly copy a disk, but these seem to be hardware dependent. SCOPY uses only standard FLEX calls, and is therefore hardware independent. This utility has been tested on my CC as well as the SWTPC system, with both 5" and 8" disk drives. All of the information necessary to write SCOPY is available in the Programmer's Guide portion of the FLEX manual.

```
 3
 4
 5                    * THIS UTILITY WILL COPY ALL THE CONTENTS OF A DISK
 6                    * IN DRIVE 0, TO A DESTINATION DISK FORMATTED WITH
 7                    * THE SAME NUMBER OF TRACKS AND SECTORS IN DRIVE 1.
 8                    *
 9                    * THE DESTINATION DISK MAY BE USED.  THE CONTENTS
10                    * WILL BE OVERWRITTEN.  IF THE DISKS ARE NOT
11                    * FORMATTED IN THE SAME WAY, AN ERROR WILL RESULT
12                    * AND THE COPY WILL BE ABORTED.
13                    * THE SPEED OVER REGULAR COPY IS ABOUT 3 TO 1
14                    *
15                    * FLEX EQUATES
16                    *
17          CD03 WARMS  EQU   $CD03
18          CD15 GETCHR EQU   $CD15
```

```
                CD18  PUTCHR  EQU   $CD18
                CD1E  PSTRNG  EQU   $CD1E
                CD24  PCRLF   EQU   $CD24
                CD3F  RPTERR  EQU   $CD3F
                CD42  GETHEX  EQU   $CD42
                CD3C  OUTHEX  EQU   $CD3C
                D406  FMS     EQU   $D406
                CB40  NFCB    EQU   $CB40
                      *
                      * VARIABLES
                      *
C100                  ORG   $C100
                      *
C100 20  06     START BRA   BEGIN
C102            CURTRK RMB   1
C103            CURSEC RMB   1
C104            LSTTRK RMB   1
C105            LSTSEC RMB   1
C106            HLFSEC RMB   1
C107            DENSTY RMB   1       'D FOR DOUBLE, 'S FOR SINGLE
C108 4F         BEGIN  CLRA
C109 1F  8B             TFR   A,DP    SET DIRECT PAGE 0
C10B BE  CB40           LDX   #NFCB   POINT AT WORKING FCB
C10E A7  03             STA   3,X     DRIVE 0
C110 86  10             LDA   #$10
C112 A7  84             STA   0,X     OPEN SIR CODE
C114 BD  D406           JSR   FMS
C117 1026 0117          LBNE  FILERR  EXIT ON FILE ERROR
C11B 86  07             LDA   #7
C11D A7  84             STA   0,X     GET INFORMATION RECORD
C11F BD  D406           JSR   FMS
C122 1026 010C          LBNE  FILERR
C126 EC  88 66          LDD   102,X   LAST TRACK AND SECTOR
C129 FD  C104           STD   LSTTRK
C12C 57                 ASRB
C12D F7  C106           STB   HLFSEC
C130 86  01             LDA   #1      CHECK DRIVE 1 FOR SAME T&S
C132 A7  03             STA   3,X
C134 86  10             LDA   #$10
C136 A7  84             STA   0,X     OPEN SIR
C138 BD  D406           JSR   FMS
C13B 1026 00F3          LBNE  FILERR
C13F 86  07             LDA   #7
C141 A7  84             STA   0,X     GET INFO RECORD
C143 BD  D406           JSR   FMS
C146 1026 00EB          LBNE  FILERR
C14A EC  88 66          LDD   102,X
C14D 1003 C104          CMPD  LSTTRK  (COMPARES BOTH TRACK AND SECTOR INFO)
C151 27  0A             BEQ   ATRK
C153 30  8D 00E1        LEAX  BADFMT,PCR
C157 BD  CD1E           JSR   PSTRNG  PRINT ERROR MESSAGE
C15A 7E  C0D3           JMP   WARMS
                      *
                      *
C15D 30  8D 00F5 DTRK   LEAX  QUESTN,PCR
C161 BD  CD1E           JSR   PSTRNG  SCRATCH DISK?
C164 BD  CD15           JSR   GETCHR
C167 81  59             CMPA  #'Y
C169 1026 00C2          LBNE  EXIT
C16D 30  8D 0100        LEAX  DENSN,PCR
C171 BD  CD1E           JSR   PSTRNG  SINGLE OR DOUBLE DENSITY
C174 BD  CD15           JSR   GETCHR
C177 B7  C107           STA   DENSTY
C17A BD  CD24           JSR   PCRLF
C17D CC  0001           LDD   #1      TRACK 0 SECTOR 1
C180 BE  CB40           LDX   #NFCB
C183 CE  0000 DTRK0     LDU   #0      START OF TRACK BUFFER
C186 6F  03             CLR   3,X     DRIVE 0
C188 FD  C102 DTRK1     STD   CURTRK
C18B ED  88 1E          STD   30,X    CURRENT POSITION
C18E 86  09             LDA   #9
C190 A7  84             STA   0,X     READ SECTOR
C192 BD  D406           JSR   FMS
C195 1026 0099          LBNE  FILERR
C199 108E C900          LDY   #NFCB+64  START OF SECTOR BUFFER
C19D EC  A1     MOVE1   LDD   0,Y++   TWO BYTES AT A TIME
C19F ED  C1             STD   ,U++
C1A1 108C C980          CMPY  #NFCB+320
C1A5 26  F6             BNE   MOVE1
C1A7 FC  C102           LDD   CURTRK
C1AA 01  00             CMPA  #0
C1AC 26  12             BNE   DTRK3   DRIVE TRK 0 IS DIFFERENT
C1AE B6  C107           LDA   DENSTY
C1B1 81  44             CMPA  #'D     ONLY FOR DOUBLE DENSITY
C1B3 26  08             BNE   DTRK2
C1B5 B6  C102           LDA   CURTRK
```

```
C1B8 F1  C106           CMPB  HLFSEC
C1BB 2C  00             BGE   WRTRK
C1BD B6  C102 RTRK2     LDA   CURTRK
C1C0 F1  C105 RTRK3     CMPB  LSTSEC
C1C3 2C  03             BGE   WRTRK
C1C5 5C                 INCB
C1C6 20  C0             BRA   RTRK1
C1C8 CE  0000 WRTRK     LDU   #0
C1CB 86  01             LDA   #1      D
C1CD A7  03             STA   3,X
C1CF B6  C102           LDA   CURTRK
C1D2 C6  01             LDB   #1      START AT BEGINNING OF TRACK
C1D4 FD  C102 WRTRK1    STD   CURTRK
C1D7 ED  88 1E          STD   30,X
C1DA 86  0A             LDA   #$0A    WRITE TRK CODE
C1DC A7  84             STA   0,X
C1DE 108E C880          LDY   #NFCB+64
C1E2 EC  C1     MOVE2   LDD   0,U++
C1E4 ED  A1             STD   0,Y++
C1E6 108C C980          CMPY  #NFCB+320
C1EA 26  F6             BNE   MOVE2
C1EC BD  D406           JSR   FMS
C1EF 1026 0D3F          LBNE  FILERR
C1F3 FC  C102           LDD   CURTRK
C1F6 81  00             CMPA  #0
C1F8 26  12             BNE   WRTRK3
C1FA B6  C107           LDA   DENSTY
C1FD 81  44             CMPA  #'D
C1FF 26  08             BNE   WRTRK2
C201 B6  C102           LDA   CURTRK
C204 F1  C106           CMPB  HLFSEC
C207 2C  0B             BGE   NXTTRK
C209 B6  C102 WRTRK2    LDA   CURTRK
C20C F1  C105 WRTRK3    CMPB  LSTSEC  FINISHED A TRACK?
C20F 2C  03             BGE   NXTTRK
C211 5C                 INCB
C212 20  C0             BRA   WRTRK1
C214 B1  C104 NXTTRK    CMPA  LSTTRK
C217 27  16             BEQ   EXIT    DONE WITH COPY
C219 34  06             PSHS  D
C21B BE  C102           LDX   #CURTRK
C21E BD  CD3C           JSR   OUTHEX
C221 BD  CD24           JSR   PCRLF
C224 35  06             PULS  D
C226 BE  CB40           LDX   #NFCB
C229 4C                 INCA          NEXT TRACK
C22A C6  01             LDB   #1      FIRST SECTOR
C22C 16  FF58           LBRA  RTRK0
C22F 7E  C0D3 EXIT      JMP   WARMS
C232 BD  CD3F FILERR    JSR   RPTERR
C235 7E  C0D3           JMP   WARMS
                      *
C238 20 44 49 53 BADFMT FCC   / DISKS MUST HAVE SAME FORMAT /,04
     C23C 4B 53 20 4D
     C240 55 53 54 20
     C244 4B 41 56 45
     C248 20 53 01 4D
     C24C 45 20 46 4F
     C250 52 4D 41 54
     C254 20 04
C256 20 53 43 52 QUESTN FCC   / SCRATCH DISK IN DRIVE 1? /,04
     C25A 41 54 43 48
     C25E 20 44 49 53
     C262 4B 20 49 4E
     C266 20 44 52 49
     C26A 56 45 20 31
     C26E 3F 20 04
C271 20 53 49 4E DENSN  FCC   \ SINGLE OR DOUBLE DENSITY DISKS IS/DI? \,04
     C275 47 4C 45 20
     C279 4F 52 20 44
     C27D 4F 55 42 4C
     C281 45 20 44 45
     C285 4E 53 49 54
     C289 59 20 44 49
     C28D 53 4B 53 20
     C291 20 53 2F 44
     C295 49 3F 20 04
                           END   START
SYMBOL TABLE:

BADFMT C238  BEGIN  C108  CURSEC C103  CURTRK C102  DENSN  C271
DENSTY C107  EXIT   C22F  FILERR C232  FMS    D406  GETCHR CD15
GETHEX CD42  HLFSEC C106  LSTSEC C105  LSTTRK C104  MOVE1  C19D
MOVE2  C1E2  NXTTRK C214  OUTHEX CD3C  PCRLF  CD24  PSTRNG CD1E
PUTCHR CD18  QUESTN C256  RPTERR CD3F  RTRK   C15D  RTRK0  C183
RTRK1  C188  RTRK2  C1BD  RTRK3  C1C0  RTRK4  C1C3  START  C100
WARMS  C0D3  WFCB   CB40  WRTRK  C1C8  WRTRK1 C1D4  WRTRK2 C209
WRTRK3 C20C
```

COLOR User Notes

ROBERT L. NAY
4429 Plantation Lane
Norcross, GA 30071

This month we'll look at a new Serial Adapter for the MX Series Printers from Epson, 'fix' a couple of 'Bugs', and look at an easy method to get super Music out of the Color Computer. I will also look at some of the Disk Operating Systems that are available, and a new one on the way. Let's get on with it!

New EPSON Serial Adapter with 4K Buffer

The EPSON MX-80 has proven to be an excellent Printer that has an outstanding record for reliability, but I have had a lot of trouble with the Serial Interface Boards they use. I have gone through 4 or 5 DIFFERENT Epson Serial Interface Boards (the 8151, with the 2K Buffer) over the past year and a half (with NO problems with the Printer itself). Some have had heat problems, some start "dropping" a character, some get flaky at higher baud rates (I usually run mine at 2400 Baud), etc. I have just installed one of the new Epson 8155 Serial Boards with a 4K Buffer (with my fingers crossed - I hope this one holds up). I thought I would pass along the set-up procedure for those of you that might be using this Board.

There are only two SWITCHES, but a BUNCH of Jumpers on this Board. It is build around an 8300 Microprocessor for the Controller, with a 24 pin 4K by 8 RAM Chip; the construction and overall physical quality is excellent. The two switches are accessible from the rear of the Printer so that they ca be changed without removing the Top Cover of the Printer. S-1 selects the Self Test feature; with it ON, the Printer prints out a notice stating that the RAM Checks OK, and then begins printing the characters and graphics blocks until you turn the Printer off. S-2 allows you to operate the Printer with a parallel interface without removing the Serial Board; this could be a real nice feature.

The Jumpers select Baud Rates, Parity, Handshaking, etc. J1 thru J6 determines the Baud Rate (300, 600, 1200, 2400, or 9600); I installed a jumper on J4 for 2400 Baud. J7, J8, and J9 select the Parity/Stop Bits; since the Color Computer does not use Parity, J7 is irrelevant, J9 is jumpered for NO Parity, and J8 is jumpered for 1 Stop Bit. J10 and J11 select Inverted or normal Hardware Handshake; jumper J11 for normal. J12, J13, and J14 select the type of Handshake; J12 for CTS Handshake, J13 for RTS Handshake, and J14 for DTR Handshake - jumper J14 for the Color Computer. (NOTE: the list on Page 4 of the Manual has J13 and J14 reversed, the Jumper Option list on Page 5 is CORRECT.) J15 selects the Handshake Point (how many characters from the Buffer End do you want to notify the Computer that you are ready for more data). I left this open so that it would call for more data at 140 characters; jumpering it waits until the buffer can take 512 more characters before it requests them (you might select this option if you had a 'spooling' capability). J16 selects the word length; jumper for 7 bit words, and open for 8 bit words. If you have a Version 1.1 BASIC ROM, leave it open; if one of the original ROMs, jumper it. J17 is jumpered if you want (and have the capability for) the Serial Test feature; you must have pins 2 and 3 of the 25 pin RS-232 Connector connected together for this to work. It would then test the serial data link -- I left this one open. J18 selects between an MX-80 and an MX-100; jumper it for and MX-80. J19 and J20 select the type of handshaking: leave them both open for the Color Computer (the other choices are X-on/X-off and ETX/ACK handshakes). In summary, S-1 is OFF, S-2 is ON, and jumpers on J4, 8, 9, 11, 14, 18, 19, and 20 for a 2400 Baud system with a Version 1.1 BASIC ROM on an MX-80. If you want to leave the Baud at the normal 600 Baud Rate of the Color Computer, jumper J2 instead of J4. (They supply extra jumper clips with the Board.)

BUG REPORTS, FIXES, ETC.

FLEX Conversion BUG REPORT and FIX

I discovered a "bug" in the DATA-COMP V2.0 FLEX Conversion when I finally got my new Epson 4K Buffer Serial Board 'up and running'; it seems that with a large Buffer on the Serial Card and a high Baud Rate, the Printer Status Check times out before the Buffer runs down far enough to need more characters. I would get a "PRINTER NOT READY" Message and the print out would continue on the Display Screen, discontinuing the Hardcopy Printout. If you run into this situation, the following "Patch" (from Steve Odneal) will eliminate the problem.

Create a File called PATCH as follows:

```
ORG $E2A7
FOB $1212
END
```

(Remember to put a 'space' in front of each line)
Now you can Assemble the file with

```
ASMB PATCH
```

which will create a PATCH.BIN File on the Work Disk. Next, change the name of FLEX.SYS on the System Disk with

```
RENAME 0.FLEX.SYS FLEX.OLD
```

which will then allow us to 'patch' FLEX.SYS with the Append Command like this:

```
APPEND 0.FLEX.OLD 1.PATCH.BIN 0.FLEX.SYS
```

Now we can try the new FLEX.SYS (which is patched) by shutting the system down and bringing it back up to get the new FLEX.SYS in operation. If the patch did not accomplish what we wanted, all we have to do is 'delete' it and rename the original File (which we called FLEX.OLD here) back to FLEX.SYS. (THIS patch works; this discussion was for those still feeling their way around with the FLEX Operating System. DO NOT destroy something that works until you are SURE that your patch is going to work OK.)

RS Disk SCRIPSIT Problem and FIX

I received a note from

Dave Pedersen
4541 Fremont Lane
Plano, Tx. 75075

reporting a problem he ran into with the new Radio Shack Disk SCRIPSIT Program. It allows you to select one of three options for Printing; 1. Print the current line, 2. Print the text in memory on the printer, or 3. Spool the text to the Disk for later printing. He is using an Okidata Printer which would ONLY work from Option 3 (Spool from Disk). He contacted Tandy and explained the problem; after some research, they came back with the answer that the Printer Driver for the first two options outputs an extremely NARROW "STOP BIT" (MUCH LESS than the half-bit specified in the RS-232-C Interface Timing requirements). Since their Printers are Edge Triggered rather than Level Triggered, they work fine and Tandy has NO INTENTIONS of providing a 'fix' to make it compatible with other Printers (or conform to any other Standard in the World -RLN-).

Well, Dave developed a "fix". He traced the problem to the DOS/BIN File, and patched it as follows:

1. LOADM"DOS/BIN from the Backup Disk
2. POKE &HEBC, &H80
3. POKE &HEBD, &H06
4. POKE &HEBE, &H12
5. SAVEM"DOS/BIN", &HE41, &HIEA0, &H1050

Now, when you RUN"DOS/BIN" the Printer works OK on all

modes.
Many thanks, Dave.

If you will send in YOUR Fixes. Patches, etc., we'll be glad to pass on the word. Maybe we can make this a regular feature.

THE COMPOSER -- A Music Compiler for the CC

Requires 16K w/ EXtended BASIC
Cass -- $24.95
Disk -- $29.95

Speech Systems
38 W 255 Deerpath Rd.
Batavia, Il. 60510
(312) 879-6880

THE COMPOSER is a Menu Driven Program which allows the user to enter, compile to Machine Language, and play Music with up to 4 different voices. It does not require any additional hardware to operate with the Color Computer. It uses the same basic algorithms as Clell Dildy's "Country Road" which appeared in this magazine several months ago, and has been used by other pubilshers in other magazines. This program allows you to enter the notes directly from sheet music through a simple, well defined procedure, compile the date to Machine Language, and play it. The 16K Cassette System allows up to 280 note groups, while the 32K Disk System allows the entry of 740 note groups.

Some of the features of this Program are:

- Four Voices.
- A unique waveshape (tone) for each voice.
- Seven full octave range.
- Full documentation and examples.
- Variable tempo to change music speed.
- Key and Octave modification possible.
- Both the Tempo and Key/Octave values may be changed as the music is playing.
- Note counter to ease debugging.
- Kaleidoscope graphics as the music plays.
- Sound Effect example and programming suggestions are provided.
- Dotted and double dotted notes supported.
- Normal, quarter, and eight note tripletts supported.
- Available in cassette and disk versions.
- User friendly menu driven software.
- The Music is saved as an independent machine language subroutine; therefore, it may be loaded and executed from other programs (e.g. BASIC).
- The compiled machine language music is in Position Independent Code (PIC) which allows the music to be loaded anywhere in memory.

The main program is written in BASIC with the following menu:

1. LOAD SOURCE (BASIC)
2. COMPILE SOURCE
3. PLAY MUSIC
4. MODIFY TEMPO
5. MODIFY KEY / OCTAVE
6. SAVE COMPILED MUSIC (ML)
7. SAVE SOURCE (BASIC)
8. END OR EDIT

The notes are entered as normal BASIC "DATA" Statements, one statement per note group, which MUST end with DATA DONE. The DATA statements begin with line number 3010, and are the only lines the user will normally change. When THE COMPOSER is first loaded, there will be a Source Program (the DATA Statements) already there. (You might went to delete these and save a 'clean' version of the program for normal use.) You can use this for experimentation, etc. Normally, you would enter your own music as DATA Statements which consist of the four-note groups (if you have less then four, add rests to replace the balance). A note may have the values of A, B, C, D, E, F, or G, for normal notes, or be followed by a # for sharp or ! for flat. A rest is indicated by RI. Following each note is the octave number, 1 to 7. Lastly, the duration for the note group is indicated by one of sixteen codes, which can be modified with a '.' for a dotted note or a ':' for a double dotted note. For example,

3010 DATA C#5G4E4B!4Q

would be a note group consisting of a C Sharp of Octave 5, a G with Octave 4, a E with Octave 4, and a B Flat with Octave 4. The note group would have a duration of a Quarter Note. Tempo can be changed with a statement such as

3070 DATA TEMPO=44

Key or Octave shifts also consist of a similar DATA Statement, where a KEY=1 would move the entire music up one Key, a KEY=-1 would move it down one Key, a KEY=12 would move it up one Octave, etc. For us non-music types, a chert is provided which shows the different notes on a scale, with duration notes, etc. Also, one line of Single Voice and one line of Four Voice Music is shown with the appropriate DATA Statements to help you get the idea. It is actually extremely easy to get the hang of it, and the Music has to be heard to be believed.

Once the DATA Statements are entered, it is then compiled by selecting Menu Option No. 2. The Program converts the DATA Statements to a Binary format for driving the Audio out to the TV Set. An extremely nice feature of this program is that this SAME MUSIC is ALSO sent to the Cassette Output at the same time, so you can get a recording by simply turning on the Recorder while it is playing through the TV. The Binary Code is in the form of a completely "Runable" Subroutine, so it can be saved with Menu Option No. 6 and called from either another BASIC Program or a Machine Language Program.

The Documentation with THE COMPOSER is excellent, and includes the DATA Statements for "King of the Road", Adeste Fideles", "Battle Hymn of the Republic", "Blown in the Wind", "Mexican Hat Dance", and "Jimmy Crack Corn". Also included is a short discussion of "Sound Effects", with a liberally commented Assembly Language Routine which can be used as a basis for developing your own Sound Effects. Finally, there are several examples of the Music supplied with the Program, and a list of 16 References for those who want to pursue the subject further.

This is another excellent example of what can be accomplished with the limited resources available with the Color Computer. When I first started writing this column I stated that this computer had an extremely powerful BASIC Operating System (and caught a little 'flak' over the statement), but I am continually amazed at the imagination that Users have when using it. I am beginning to suspect that ANYTHING is possible with this System. If you went to just ENJOY this Computer, or impress your friends with it's capability, THE COMPOSER is a must.

STAR-DOS -- A Disk Operating System for the CC

Requires a Disk System
Disk -- $49.95

Star Kits
P. . Box 209
Mt. Kisco, N.Y. 10549
(914) 241-0287

We have been following the development of Disk Operating Systems for the Color Computer since the Column first began, and with considerable interest. Obviously, a good DOS is required to allow efficient use of ANY Computer System, and this has been the primary thrust in making the Color Computer a WORKING machine.

One of the first Disk Systems that became available for the Color Computer was the Tellgrass Controller with a DOS written, and still supported, by Bill Vergona of Cer-Comp (his Tape Based Editor and Assembler are still the best I have seen for use with Tape Systems). Bill is now selling the Controller and his DOS as a complete package. His DOS provides the best interface with the Radio Shack BASIC Operating System I have seen, but the System has a fatal flaw, which prevents it's becoming a force in the market place. The Cer-Comp System uses Hard Sectored Disks, which makes it incompatible with ANYTHING else on

the market. This system will find some use in very specialized areas, where the user is only interested in his OWN use of the Computer, and is not the least bit interested in what he or anyono else is doing with the Color Computer.

The next System to appear was Exatron's Disk Controller and DOS. This was the first System used by Data Comp for getting the FLEX DOS running on the Color Computer, but it fell by the wayside due to Exatron's lack of support of the Controller in developing Double Density, Double Sided capability.

About a year ago, Radio Shack's long-awaited Disk Controller and Disk Operating System appeared, and after a short 'debugging' period, has become THE Disk Controller for the Color Computer (as was expected, for many reasons). Again obviously, anyone interested in realizing the potential of the market place, and anyone interested in making the most of using the Color Computer, will be using the Radio Shack Controller, or at least something COMPATIBLE with it. While the Color Disk DRIVES are not 'High Quality' items (in my opinion), the Controller has been a good working unit, is fairly 'adaptable' to other Systems, and is a good trade-off between Cost and Quality. The Radio Shack Disk Operating System supplied with the Disk System is a good, functional, DOS, which again is an excellent marriage of the various trade-offs which occur in designing a 'marketable' product. It provides the necessary fundamental operations, and the BASIC Operating System supplied with it is simple but extremely powerful (as is the whole BASIC Operating System supplied by Radio Shack). The Disk Controller has been used as the basis for all of the FLEX adaptations, and in conjunction with the MC6883 SAM and it's capabilities (and restrictions), has allowed an easy marriage to the FLEX Disk Operating System which is used by the majority of the 68xx Computer System Users, including it's Multi Track, Multi Drive, and Multi Side Disk Operations. The use of the FLEX Operating System has opened the door for the 'Serious' Color Computer User to a vast amount of powerful, already debugged, software; and the few Programs which need modification to work with these conversions are being converted as rapidly as possible.

So where does STAR-DOS fit into this picture? Pete Stark, the man behind Star-Kits, found a gap between the Radio Shack BASIC Operating System and the powerful FLEX Operating System. The limitation with the Radio Shack System is that it ONLY operates within the BASIC Language and Operating System; it is not convenient to use with non-BASIC Programs. FLEX, while extremely powerful, DOES require that the Color Computer be modified, which will void the Radio Shack warranty if the Computer is less than 90 days old, and you either have to run Radio Shack BASIC, OR run FLEX; the two can not coexist WHILE RUNNING A PROGRAM. STAR-DOS is being developed to fit in between these two Disk Operating Systems. It can best be described as a "baby FLEX" Operating System; it has one of the strongest features of FLEX in the ease with which Machine Language Programs can be interfaced with the DOS (in fact, many FLEX Programs can be interfaced easily by simply changing the System "Equates"), while providing some of the power, consistent command structure, and ease of operation that has made FLEX so dominant in the industry. I have just received an initial copy of STAR-DOS, and will put off a full report till a later Column, but one thing is for certain; if Pete wrote it, it will be a good, solid piece of Software. Only time will tell whether there is enough of a gap for this DOS to find a "home" in the industry; if that gap does exist, this will be the Software to fill it.

HEX ♣

by

T. M. Baleshta
42 Herrington Court
Nepean, Ontario K2H 5N7
(C) 08 April 1982

A HEX Dump and Load Utility Program, written in BASIC, for the TRS-80C Color Computer.

I wrote this program when I first obtained my TRS-80C and needed a HEX Dump and Load utility. Since then I made the program more user friendly. There are much faster machine language programs available, but HEX+ should suit the BASIC programmer. This program will run on COLOR BASIC as well as EXTENDED COLOR BASIC machines.

When the program is first run a hex dump or load(backslash)? prompt will appear. Use the following commands:

'D' (Dump) <cr> -- The program will prompt for start and end addresses which must be given in 4 place hexadecimal, i.e., presented in xxxx form from 0000 to FFFF. The program will echo the input hex address as well as print the equivalent decimal value. An end address lower in value than the start address will result in an end address reprompt. A partial dump from 0000 to 005F is given in Table 1 as an example of the Line Printer format. Video printing is restricted to 8 hex-byte-character lines.

'L' (Load) <cr> -- Address prompting is the same as the dump routine. Input hexadecimal values will be displayed (video only) as 8 hex-byte-character lines. The load program does not enter in a reverse address direction so re-entrys or corrections should be made using the T Command.

'R' (Dump Reset) -- The program will re-initialize the dump routine with address prompts regardless of what mode the program is in.

'S' (Program Reset) -- The program itself will be re-initialized with the dump/load prompt displayed.

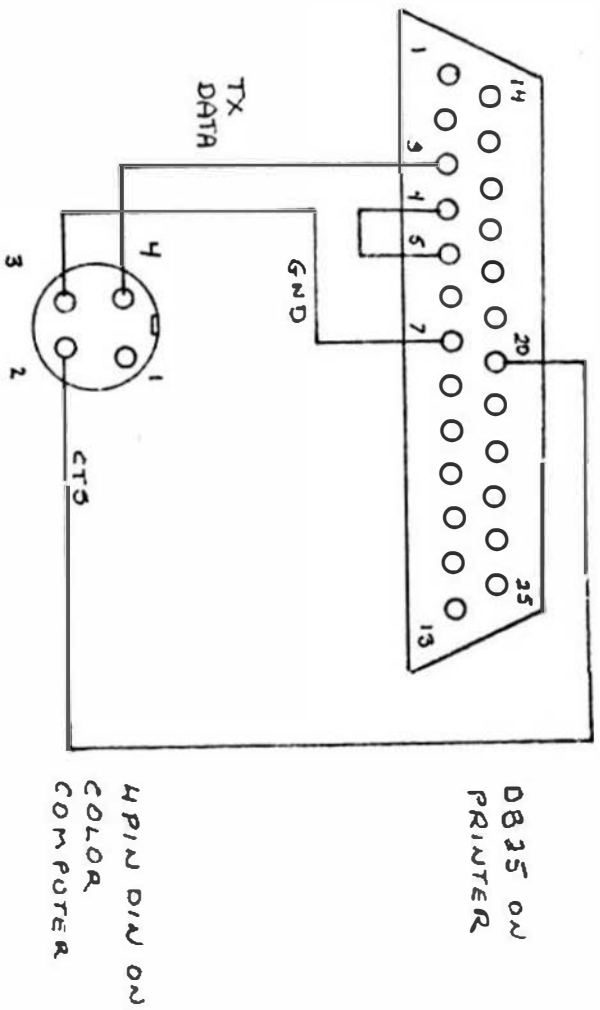
'T' (Load Reset) -- This command is similar to the R Command except a load reset is effected.

'X' (Exit) -- Causes program exit back to the color basic monitor.

When a dump or load routine has ended, the program is re-started with the dump/load prompt. Of course, you can use the R, S, T, and X commands anytime during a dump or load program.

A Line Printer routine for 300 Baud Printing is provided (Delete Lines 110 and 120 if you use the normal Color Computer Printer routines --- RLN). Connections to the Color Computer's DIN printer port and a typical DB25 Line Printer connetor are shown in Fig. 1. If you do not wish to use the Line Printer routine, eliminate lines 830, 840, 890 and 1600 (all the lines with the PRINT #-2 statement). Failure to eliminate these lines without the Line Printer handshake to the Color Computer will cause Color BASIC to hang.

FIGURE 1. PRINTER CONNECTIONS FROM COLOR COMPUTER TO STANDARD DB25 CONNECTOR




```
10 ' HEX+
20 ' A HEX DUMP AND LOAD PROGRAM
30 ' FOR THE TRS-80C COLOR COMPUTER
40 ' BY T M BALESHTA
50 ' 42 HERRINGTON COURT
60 ' NEPEAN ONTARIO CANADA  K2H 5N7
70 ' 613-S28-7212
80 ' (C) 8 APRIL 1982
90 '
100 ' MAIN PROGRAM
110 POKE 150,180 'SET 300 BAUD (set for your sys)
120 POKE 151,64 '.29 SEC DELAY
130 PRINT
140 INPUT "HEX DUMP OR LOAD";I$
150 IF I$="D" THEN 210 'DUMP
160 IF I$="L" THEN 340 'LOAD
170 IF I$="X" THEN END 'EXIT
180 GOTO 140
190 '
200 ' PROGRAM FOR DUMP
210 PRINT:PRINT "HEX DUMP"
220 GOSUB 460
230 IF A$="R" GOTO 210 'R-DUMP
240 IF A$="S" GOTO 130 'R-PROG
250 IF A$="T" GOTO 340 'R-LOAD
260 GOSUB 810
270 IF I$="R" GOTO 210
280 IF I$="S" GOTO 130
290 IF I$="T" GOTO 340
300 PRINT
310 GOTO 130
320 '
330 ' PROGRAM FOR LOAD
340 PRINT:PRINT "HEX LOAD"
350 GOSUB 460
360 IF A$="R" GOTO 210
370 IF A$="S" GOTO 130
380 IF A$="T" GOTO 340
390 GOSUB 620
400 IF A$="R" GOTO 210
410 IF A$="S" OOTO 130
420 IF A$="T" GOTO 340
430 GOTO 130
440 '
450 ' ADDRESSING SUBROUTINE
460 PRINT "START ADDRESS  ?";
470 GOSUB 1040
480 IF A$>"F" OOTO 590 ELSE 490
490 Y=D
500 PRINT Y
510 PRINT "END ADDRESS    ?";
520 GOSUB 1040
530 IF A$>"F" GOTO 590 ELSE 540
540 Z=D
550 IF Z<Y GOTO 560 ELSE 580
560 PRINT
570 GOTO 510
580 PRINT Z
590 RETURN
600 '
610 ' MEMORY LOAD SUBROUTINE
620 F=0
630 PRINT TAB(0) HEX$(Y);
640 C=1:E=0
650 GOSUB 1150
660 IF A$>"F" GOTO 780 ELSE 670
670 E=E+B*16^C
680 PRINT TAB(8) A$;
690 IF C=0 THEN 720
700 C=C-1
710 GOTO 650
720 POKE Y,E
730 PRINT " ";
740 F=F+1:Y=Y+1
750 IF Y=Z+1 THEN 780
760 IF F=16 THEN 620
770 GOTO 640
780 RETURN
790 '
800 ' MEMORY DUMP SUBROUTINE
810 P=0:PRINT
820 PRINT TAB(0) HEX$(Y);
830 PRINT #-2,CHR$(13);
840 PRINT #-2,TAB(0) HEX$(Y);TAB(8) CHR$(0);
850 LET A=PEEK(Y)
860 IF A<16 GOSUB 1430
870 IF Q=1 THEN 900
880 PRINT TAB(8) HEX$(A);" ";
890 PRINT #-2,HEX$(A);" ";
900 Y=Y+1
910 Q=0
920 IF Y=Z+1 THEN 1010
930 P=P+1
940 IF P=16 THEN 810
950 I$=INDEY$
960 IF I$="R" THEN 1010
970 IF I$="S" THEN 1010
980 IF I$="T" THEN 1010
990 IF I$="X" THEN END
1000 GOTO 850
1010 RETURN
1020 '
1030 ' ADDRESS VALUE SUBROUTINE
1040 C=3:D=0
1050 GOSUB 1150
1060 IF A$>"F" GOTO 1120 ELSE 1070
1070 D=D+B*16^C
1080 PRINT TAB(16) A$;
1090 IF C=S THEN 1120
1100 C=C-1
1110 GOTO 1050
1120 RETURN
1130 '
1140 ' INPUT VALUES SUBROUTINE
1150 A$=INDEY$
1160 IF A$="" THEN 1150
1170 IF A$="0" THEN B=0
1180 IF A$="1" THEN B=1
1190 IF A$="2" THEN B=2
1200 IF A$="3" THEN B=3
1210 IF A$="4" THEN B=4
```

```
1220 IF A$="5" THEN B=5
1230 IF A$="6" THEN B=6
1240 IF A$="7" THEN B=7
1250 IF A$="8" THEN B=8
1260 IF A$="9" THEN B=9
1270 IF A$="A" THEN B=10
1280 IF A$="B" THEN B=11
1290 IF A$="C" THEN B=12
1300 IF A$="D" THEN B=13
1310 IF A$="E" THEN B=14
1320 IF A$="F" THEN B=15
1330 IF A$="R" THEN 1400
1340 IF A$="S" THEN 1400
1350 IF A$="T" THEN 1400
1360 IF A$="X" THEN END
1370 IF A$>="0" THEN 1150
1380 IF A$<"0" THEN 1150
1390 IF A$>"9" AND A$<"A" THEN GOTO 1150
1400 RETURN
1410 '
1420 ' "0-F" VALUES SUBROUTINE
1430 IF A=0 THEN H$="00"
1440 IF A=1 THEN H$="01"
1450 IF A=2 THEN H$="02"
1460 IF A=3 THEN H$="03"
1470 IF A=4 THEN H$="04"
1480 IF A=5 THEN H$="05"
1490 IF A=6 THEN H$="06"
1500 IF A=7 THEN H$="07"
1510 IF A=8 THEN H$="08"
1520 IF A=9 THEN H$="09"
1530 IF A=10 THEN H$="0A"
1540 IF A=11 THEN H$="0B"
1550 IF A=12 THEN H$="0C"
1560 IF A=13 THEN H$="0D"
1570 IF A=14 THEN H$="0E"
1580 IF A=15 THEN H$="0F"
1590 PRINT TAB(8) H$;" ";
1600 PRINT #-2,H$;" ";
1610 Q=1
1620 RETURN
```

```
;SP 2
;CE
TABLE 1 --- Sample Printer Dump
;SP
0    23 00 22 00 00 00 00 00 00 00 02 01 A9 01 A4 7F
10   27 00 00 78 00 00 00 7F 1E 1E 01 29 0C 29 59 29
20   59 7F 36 7F F5 7F F6 7F FE 00 AA 03 52 00 00 23
30   AC 00 00 1E 00 02 DE 59 00 29 31 29 46 00 00 00
40   00 20 00 00 00 00 00 00 23 AD 00 81 00 00 01 A9 87
50   00 00 00 53 28 00 01 00 24 98 00 00 00 00 00 29
60   00 00 00 00 24 9A 29 04 03 52 10 70 2D 84 00 00
70   00 55 80 C0 7F FE 00 00 00 00 01 DA FF 00 29 0C
```

"C" User Notes

Norm Commo
5 Pryor Road
Natick, MA 01760

C NOTES 9

This month I would like to finish up what has been a slowly dwindling, somewhat tutorial, description of the C language by covering unions and compiler directives. I will leave some the more obscure features and newer extensions till later. As I have said earlier, if you are not running full blown C then you most likely won't need them with your compiler.

UNIONS

The union is a construct in C that allows you to use one variable for storing different data types. When the compiler encounters a union, it allocates storage necessary to hold the largest of its members. The declaration of a union would be

```
union tag   {
            .
            .
            data declarations;
            .
            .
            } u_name;
```

As you can see, it is very similar to the struct's declaration. In fact, the components act similarly.

tag
This is a "tag" name given to the union since you may have other unions with different formats. It also allows you to use the union as a template for other declarations.

data declarations
These are the declarations of all the possible data types that the union will hold at some time or other. They may be simple types such as char's and int's, or they may be complex types such as arrays, struct's and other unions.

u_name
This is the name of an actual variable of the type "union tag". It causes the necessary storage to be allocated for the variable.

Both tag and u_name are optional components as long as at least one is used. All the examples that I gave last month can be judiciously applied to unions. The main thing to remember about unions, as compared to struct's, is that only one of the data types mentioned in the declaration will be stored in the union at any given time. Furthermore, it is your responsibility as the programmer to keep track of what's actually in there at any given time. The compiler will just make sure that the union has enough storage allocated to hold largest data type declared within it.

Like struct's, unions may be global, local or static and are referred to in a program in one of two ways. If the union is an actual variable, then the syntax is

```
union_variable_name.member_name
```

If you are working with the union indirectly via a pointer, then the syntax is

```
union_pointer_name->member_name
```

Well, so much for dry theory, let's look at a few examples.

```
union   {
        int i;
        char *stg[5];
        };
```

Here, the size of the union would be 5 bytes since that's the size of its largest member, stg.

```
union   {
        int i;
        char c;
        float f;
        };
```

The size of this union would be the size of the float, which is normally 4 bytes although strictly speaking it is compiler dependant.

The next example is a little more practical. It is a fragment of a structure that is used to represent a FLEX file control block, and is taken from a header file that comes with the Introl compiler.

```
.
.
union   {
        /* new name of renamed file */
        char    new_name[11];
```

```
    struct  {
            char    scratch[6]; /* fill */
            char    flag; /* text flag  */
            } c;
    } s;
```

This is a very interesting example because it shows a structure, "c", declared within a union, "s", which in turn is declared inside another struct, which happens to be tagged "frw". Whew! If you have done much FLEX assembler programming involving file I/O then you know that the new name for a file rename is stored as 11 bytes in the fcb. That same space is also used for scratch and the text compression flag when if the fcb is used for a file that is open for read or write.

The notion of "a filename OR scratch space and a flag" dictates using a union which is what Intrel did. How to use it in a program? Let's lay down the assumption that we will be using a pointer to the struct frw called file_blk. If we want to refer to the filename for a rename, then the syntax is

```
file_blk->s.filename
```

Similarly, if we want to get at the space compression flag the syntax is

```
file_blk->s.c.flag
```

An important fact is pointed out by this example. Since the reference to the overall structure is via the pointer file_blk, top level members are accessed using "->" but once inside we use the "." linkage to reference nested members. This is true for both sturct's and unions.

If we were dealing with a variable "fcb" of the type frw, then the two references would have been

```
fcb.s.filename
```

and

```
fcb.s.c.flag
```

You can easily see that while struct's and unions are very powerful, they can be carried to extremes without too much effort. Consider a nesting which requires the following reference

```
top.nest1.nest2.nest3.nest4.nest5.probably_lost
```

The compiler would sit there and crank it all out but could we (mere humans) really keep track ?

COMPILER DIRECTIVES

The C language has a number of compiler directives built into it. The two most common, #include and #define, are available in every compiler that I've ever seen. The number actually available to you will depend on your particular compiler. Here is a list of the others with a brief description of each.

#include <filename>
This causes the named file to be included in the compile at that point in the source. The form for the filename will be either <filename> or "filename" again depending on the compiler.

#define symbol string
This is a substitution directive, or macro. It causes every occurrence of symbol to be replaced by string.

#define symbol(arg1, arg2... argn) string
This is a macro with substitution. As before, every occurrence of symbol is replaced with string, except the arguments are "passed" or substituted into the string.

#if expression
If the constant expression evaluates to some non-zero value then code from the #if to an #endif or #else will be compiled.

#ifdef symbol
If the symbol is defined (by the preprocessor) then code will be compiled, as with #if.

#ifndef symbol
If the symbol is not defined, then compilation of following code will continue as with #if.

#endif
Ends a conditional compilation.

#else
Allows an alternate code body to be compiled within a conditional compilation. It must occur between one of the #if's and a #endif.

#line constant identifier
A diagnostic tool that let's the compiler, for errors, believe that the next line has the line number "constant" and comes from the file "identifier". identifier is optional.

sizeof (object)
or
sizeof (expression)
This will force the compiler to evaluate a constant based on the expression or the size of a particular object such as a data type or variable. A most handy tool.

typedef data-type new_name;
This will cause the compiler to bind new_name to data-type thereby making them synonymous. In simple cases, it is similar to #define. The difference is that #define merely swaps every occurrence new_name for data type, where as the typedef creates a new data type. This allows new_name to be used successfully in certain cases (such as hairy declarations) where it could not be had it been #defined.

Let's look at some code fragments that illustrate some of these directives. I am in the process of coding up a fairly elaborate program for programming eproms. Of necessity, there are a number of global variables that reference the prom buffer. I am using the Intrel compiler. This compiler requires that the definitions and declarations of globals occur in the same order for all the files used in the program. In a case like this, the variables are declared in one file (that is their storage is actually allocated) and defined (or imported) in all the others.

One way to handle this would be to put all the declarations and definitions in the front of each file. The pitfall of this approach is that any change or additions to the globals must be carried across all the source files. A better way to handle this is to use a header file that gets #included in each of the files. What I used was the #ifdef with two sections, the first depended on the symbol MAIN being defined. If it was, then the variables were declared. Otherwise, they were imported (defined). This is shown below.

```
#ifdef MAIN    /* global storage */
   char *promram;
   int curradd, baseadd;
    int windtop, windbot;
   char *filstg; *filstat;
   FILE *fileid;
   int promtype, psize;
   BOOL screenchanged;
#else       /* the global externs */
   extern char *promram;
```

```
        extern int curradd, baseadd;
        extern int windtop, windbot;
        extern char *filstg; *filstat;
        extern FILE *fileid;
        extern int promtype, psize;
        extern BOOL screenchanged;
    #endif
```

In the first source file, which actually did contain main(), I used the following two lines

```
#define MAIN 1
#include "prom.h"
```

which caused the compiler to allocate storage for the variables. For all the other files, I used only

```
#include "prom.h"
```

In which case the compiler merely knew what type each variable was. Now, when I need to make an update I do it only in one file. It is still necessary to recompile all the files after a significant change, but I don't have to modify them.
And the since the declaration and definition sections are physically close together in the header file, the chances for errors is greatly diminished.

I recently completed two mailing label programs. The code in these programs constantly referred to chunks of memory that contained a mailing label consisting of 4 lines of 35 characters. This was a natural for the typedef directive

```
#define LABLEN 36   /* 35 plus a NULL */
#define LABHGT 4
/* label needs a "safety" line */
typedef char label[LABHGT+1][LABLEN];
```

By typedef'ing label, I could use it like as a "regular" data type. For example, I needed a function that would clear a label buffer to spaces. It looked like

```
clr_lab(lab)
    char *lab;
    {
    int i;

    for (i = 0; i < sizeof(label); i++)
        *lab = ' ';
    }
```

clr_lab() uses a pointer to char's for efficiency. The data type "label" is combined with sizeof() to determine the number of iterations. This is an example of how you typically use the sizeof() directive.

One of the programs is used to save paper by printing a file of labels in three columns of ten per page. It was also convenient to have the columns sorted in dictionary fashion, such that the first name of the second column followed the last name of the first column etc. However, the file was arranged in strict alphabetical order. It was necessary to create a "page" buffer that would hold 30 labels. That way, I could read them in sequentially filling in column one, then column two and finally column three. I would then print them out "across the page". Let's take a look at the page buffer declaration and the function that reads in the label from a file.

The page buffer is a two dimensional array of three by ten labels declared as

```
#define LABS_PER_COL 10
#define COLS_PER_PAGE 3
label labpage[LABS_PER_COL][COLS_PER_PAGE];
```

Any particular character in any label on the page can be addressed as a four dimensional array, for example

```
labpage[2][5][3][25]
```

refers to the 25th character of the third line of the fifth label in the second column. You may omit indices, starting from the right. They will be assumed as zero. So we could have

```
labpage[2][5][3]
```

which is the first character on the third line etc, or perhaps

```
labpage[2][5]
```

which would be the first character of the first line etc.

Next we have the function getlab() which reads in labels from a file ignoring leading blank lines.

```
getlab(lab,col, fileid)
    int lab, col;
    FILE fileid;
    {
    char line[LABLEN+1];
    int i;
    BOOL inlabel;

    i = inlabel = FALSE;
    FOREVER
        {
        if (fgets(line,LABLEN+1,fileid) == NULL)
            return(FALSE);

        if (line[0] == '\n')
            if (inlabel)
                break;
            else
                continue;
        else
            {
            labcpy(&labpage[lab][col][i++],line);
            inlabel = TRUE;
            }
        }
    return(TRUE);
    }
```

getlab() is passed integer indices for the current label of the page buffer to be filled from the file. It should be pointed out that labcpy() is nothing more than a slightly more intelligent strcpy() which loads a line of text from the file into one line of the label.

WRAPUP

I got a call from Harold Harkness of Word's Worth the other night. They are moving right along and will have a real relocating assembler soon. I have also received a copy of their "Middle C" complier. I had hoped to have a preliminary review of that with this column but circumstances blew those plans away. I will make a hard effort to have it for the next column.

By now, which is early November, Introl should be out with floats and longs. I await that version with bated breath.

I would like to thanks the readers that have sent in ideas and desires. It really has helped me get a feel for what some of you would like to see. I had an enlightening talk with a 68 Micro reader and former work associate. He pointed out that C is still fairly new to the 68XX scene and that might account for a lack of response. This view was confirmed in a letter from a reader in New Vancouver, BC. He said "Perhaps there are many like me who are new to 'C'...".

I can only add that if you don't have any thing to contribute yet, but know what you want to see then let me know.

I feel fairly sure that many of you are using the "Small C" derivatives because of their lower price. Don't fret, I haven't forgotten you. I use the Introl compiler because it's the biggest, easiest to use compiler that I own and it allowed me to illustrate some of C's nicer features. Now that the tutorial is done, I hope to go over many of the things we have covered again in greater depth using the "Small C" subset that is contained in the Word's Worth compiler.

The equipment here is homebrew which limits the operating systems that I can run to OS9 level 1 and FLEX. I would really like to see some in depth reviews of the compilers from TSC and SWTP. Another reader, Paul Walkin, has the FLEX Telecon compiler and I will be working with him to get a more thorough review of that product since I have heard some rumblings about it.

Dual Processor Conversion for SS-50

by Dr. Samuel I. Green
13052 Ferntrails Lane
Creve Coeur, MO 63141
4 June 1982

My basic system consists of a SWTP SS-50 box and power supply with a choice of MP-A or MP-A2 6800 CPU boards, a Hazelwood 64K RAM board, and several I/O boards addressed at $8000 in a standard MIKBUG compatible arrangement. I have a lot of system software specially tailored to my needs, so that the 6800 system must be kept operational.

I purchased the MP-09A 6809 CPU board recently. Bringing the new CPU up is relatively simple. The old CPU board is removed, and the new CPU board is installed. The address of the I/O is changed from $8000 to $E000. (I did this by soldering together the two I/O select lines, pins 9 and 15 at the socket of IC 6, and bending out the pin of the deselected address on the 74LS138 address decoder when it is plugged in.) When the computer is turned on, S-BUG will greet you and tell you how much memory is available. Since the addressing is still set up for a 6800, there will be 52K, but if the 4K block at $8000 is enabled and the computer is reset, it will come up with 56K. This is the practical limit, since 4K is dedicated to I/O and 4K is dedicated to the ROM monitor. NMI and M.RESET lines still have to be removed from the bus and routed to front panel switches through a new connector, since these bus lines have been redefined.

After reconfiguring the system for the different processors a dozen times, I began looking for a better way. There are two major drawbacks to the SWTP SS-50 bus which go back to its origins as an inexpensive hobbyist implementation. First, the motherboard plugs and sockets are tin plated which means that they don't stand up well to repeated insertions and removals. Secondly, there is no provision for card guides. The first problem has been alleviated by using gold plated connectors on the newer premium SS-50 boxes like those from GIMIX and Hazelwood. The second problem has been aggravated by spacing boards closer together in those same premium boxes.

Both the 6800 and 6809 CPU boards can be put on the bus at the same time if one board at a time is put in the halt mode and a few remaining interfering signals are dealt with. I chose to modify the MP-09A as little as possible since it was newest. I elected to modify the MP-A first, since it was expendable. When this was successful, I modified the MP-A2, since it is much more versatile.

Generally the bus lines which would conflict are tri-stated (floated) by the bus-available signal from each processor when the HALT line goes low. The exceptions are the Baud Rate lines, RESET, and the clock signal E (phase 2). One board is selected to provide the Baud Rates all the time and the other is disabled. The RESET and E outputs are output through tri-statable buffers in all CPU boards. In the case of the MP-A they are already tri-stated by BA (bus-available). On the MP-A2 and the MP-09A the buffer control pins must be ungrounded and tied to BA.

On both CPU boards the HALT signal is clocked into the CPU through a D-Type Flip-Flop. On one board I simply changed the HALT to the CPU from the Q output to the Q (inverted) output, so that when one CPU was halted the other would be running. I raise the HALT line high or low with a front panel switch to change processors. The penalty for this is that the halt line is no longer available for DMA applications.

Attention must also be paid to the state of the bus buffers on the HALTed CPU board. All three CPU boards reverse the direction with respect to the R/W signal when they are halted so that control can revert to an external DMA controller. The MP-A must be disabled in this mode since its bus buffers are not tri-stated and end up pointing onto the bus during HALT. The MP-A2 must also be disabled in this mode since it would be enabled at $A000 and $E000 as well as $F000. The MP-09A need not be disabled since it would only point onto the bus at $F000 when the MP-A or MP-A2 would not be looking onto the bus. However, the conflict between bus buffers is undesirable and seems to have caused some intermittant problems when I was programming EPROMS. Therefore, after six weeks of otherwise reliable operation, I disabled the MP-09A bus buffers when that board was halted.

NMI and M.RESET lines must be removed from the bus on the MP-A or MP-A2 since these lines are redefined for the 6809.

Finally, the I/O addresses between the 6800 and 6809 need to be resolved. I worked out three different ways to switch the adresses using the HALT signal without running high speed decoding signals to the front

panal, but I haven't implemented any of these yet. Instead I simply changed one byte in S-BUG as suggested by SWTP and left the I/O addressed at $8000. When S-BUG does its initialization following power-on-reset, the dynamic address translator (DAT) is configured to shift the address logically to $E000. SWTP says this change in S-BUG "costs the user an additional 16K of memory capability", but my system came up with 52K. I believe this is due to the fact that the Hazelwood memory is deselectable in 4K blocks while SWTP assumed larger blocks. The memory can be brought up to 56K by enabling the block at $E000 and moving it to $8000 with a manual entry to the DAT.

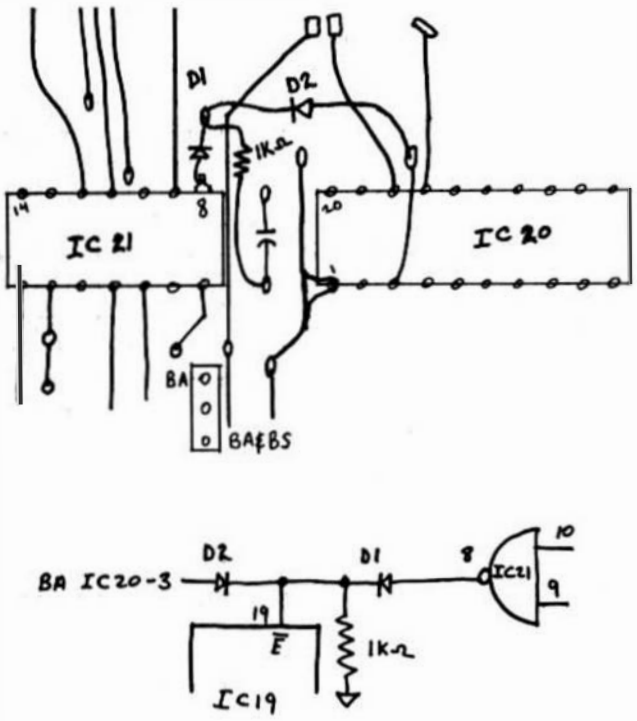


Figure 1 OR-GATE composed of D1, D2, and 1KΩ added to disable data bus buffers when BA high (MP-09A)

These modifications include the ideas of Bill Vergona ('68' Micro Journal, June, 1980) and Rudolf Reuter ('68' Micro Journal, February, 1981). Mistakes are corrected, improvements are added, and modifications to the MP-09A are minimized.

Following are the steps of my dual processor conversion.

Changes to the MP-A 6800 CPU board:

1. Disable phase 1 clock from the bus by unplugging IC 15 pin 5 (8T98).
2. Disconnect M.RESET from the bus by unsoldering the bus side of capacitor C8. Add 1 kohm pullup to capacitor and connect to separate connector.
3. Change polarity of HALT. Unplug IC 20 pin 9. Tie pin 9 to pin 8 at socket of IC 20.
4. Open IC 12 pin 12 so that buffers will point in when HALTed.
5. Open all baud rate lines by bending out IC 14 pins 2, 4, 6, 8, and 10. The last gate is required to provide CPU clock signals.
6. Cut NMI trace to bus and bring to new connector.
7. I added an M.RESET and NMI Molex connector like the one on the MP-09A board and connected them in parallel.
8. Since the other board will supply baud rates, the crystal can be changed to 2 MHz to get a true 1 MHz clock.

Changes to the MP-09A 6809 CPU board:

1. Tri-state RESET and E and Q by removing IC 23 pin 19 from its socket. Solder a wire from the removed pin to jumper pin 3S.
2. 2S the baud rates with a jumper so they're always on.
3. Remove BA from the bus by unplugging the jumper.
4. Disable the data bus buffers during HALT by adding a diode gate as in figure 1. Remove IC 21 pin 8 from the socket and solder diode D1 between the removed pin and the trace from IC 21 socket pin 8. Solder diode D2 between pin 3 of IC 20 and the same trace from IC 21 socket pin 8. Plated through holes are conveniently located for mounting the diodes. Add a 1K ohm resistor from IC 19 pin 19 to ground.
5. Change S-BUG to move $8000 I/O to $E000 by changing location FF79 from F1 to F7.

Changes to the MP-A2 6800 CPU board:

1. Tristate RESET and E (phase 2 clock) by removing IC 11 pin 15 from its socket. Solder a short wire from the removed pin to pin 1 across the top of IC 11.
2. Disconnect M.RESET from the bus by cutting the trace to the bus. Add 1 kohm pullup resistor to M.RESET and connect to a separate connector.
3. Change polarity of HALT. Unplug IC 12 pin 5. Tie pin 5 to pin 6 at socket of IC 12.
4. Disable the data bus buffers during HALT by unplugging pins 12 of IC 17 and IC 18. Tie each pin 12 to pin 2 by soldering short wires across to top of both IC 17 and IC 18.
5. Disable all baud rate lines by unplugging IC 10. IC 9 may also be unplugged.

6. Cut NMI trace to bus and bring to new connector.
7. I added an M.RESET and NMI Molex connector like the one on the MP-09A board and connected them in parallel.

Changes to the MP-B2 Motherboard:

1. Add a pull up to UD 2 which is FIRQ for the MP-09A.
2. Add a front panel switch to toggle the HALT line to switch between processors. High will select the 6809, and low will select the 6800.
3. Add a front panel switch and connect to NMI inputs of both CPU boards. This is optional since neither SWTBUG nor S-BUG support the ABORT function as does MUMBUG. The simple change to S-BUG submitted by D.R. Gaskell ('68' Micro Journal, August, 1981) adds the ABORT function and is well worth incorporating. The debounce circuit is already incorporated on the MP-09A board.

OS9-EPROM

G. A. R. TROLLOPE
466 OASWALLEN DRIVE
WEST CHESTER, PA 19380

Certainly one of the more attractive features of using OS9/BASIC09 for process control is that both the system routines and any programs written in BASIC09 can eventually placed in E-PROM, minimising the dependence upon mechanical devices like floppy discs. But, it is one thing for the routines to be romable, it is another to get them into E-PROM. I hope that this piece will help you to understand how I have achieved this.

The first thing you need ofcourse is some sort of prom burner. The job has got an awful lot simpler since 1702A's-you don't even have to switch high voltages. So, to program 2716's, one pin is tied to 25volts, and the actual programming is done by strobing a different pin that operates at 5v logic levels. Now I already owned a 2708 programmer that I had wire-wrapped some years ago. I was just on the point of tearing it apart, when it occurred to me that I could put a second socket on the (SS30) board, and wire it up in parallel to the first for the 2716's. The circuit diagram is shown in fig 1. In order to get by with only one Parallel Interface chip, I latch the three most significant address bits into a 7475 latch - the software depends upon 'doing it my way'. But, looking ahead to 2732's and 2764's, it would be better to use a 6 or 8 bit latch (maybe 2 7475's), and to leave enough room for these other sockets. Just make sure PA7 is latched to /G, output enable.

The device driver is shown in fig 2. This is a routine to be called from BASIC09 with three parameters, all INTEGERs. The first is a single address in the 2716 to program, the second is the data to be programmed, and the third is the data read back from the PROM after programming. When the PROM has been erased, all bits are converted to 1's, so that the driver does not attempt to program the data $FF, although it does read the PROM. This may be used to check that the PROM was correctly erased, or may be used to obtain a listing of the PROM.

In operation, the driver first programs both sides of the PIA to be outputs. It picks up the address, strobing the most significant bits into the latch using the CA2 control line. /G is raised at the same time. The other address bits are placed into the A side of the PIA, and the data into the B side. After stopping the OS9 clock, the PROM is programmed by raising the CB2 control line for 50 milliseconds. A simple timing loop is used, which assumes a 1 MHz clock- if your clock is different, change the loop counter. /G is lowered, the address written back, the B side converted to input, and the data read and reported.

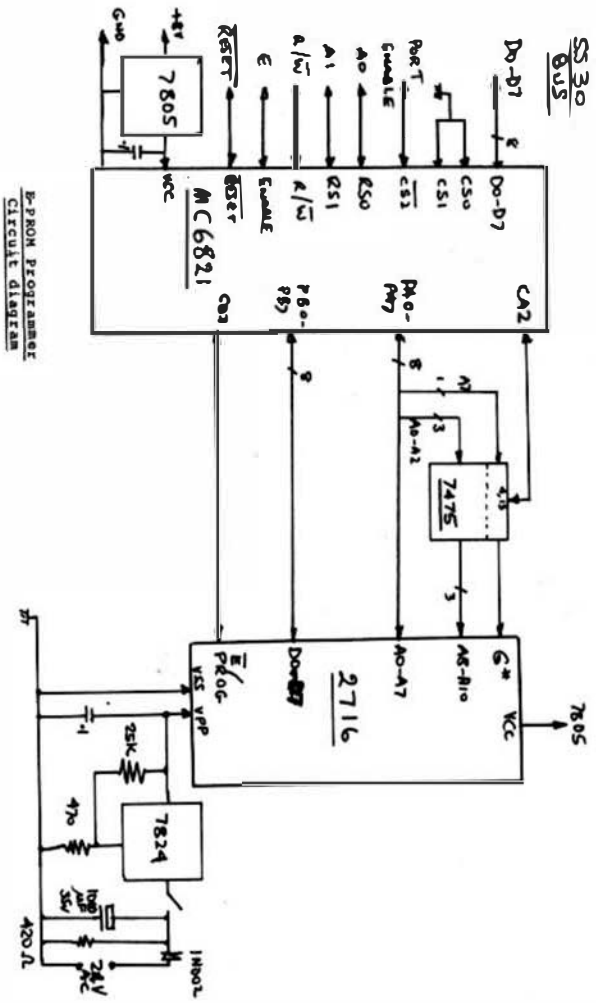
If you want to use your own programmer, the driver should not require too much modification to be compatible with the BASIC09 control

program, given in Fig. 3. In General terms, this reads the programs to be placed into E-PROM from the disc, so that all such programs must first be SAVEd. This includes device descriptors such as D0 and D1, and modules like SYSGO and INIT. Programs like SHELL and DIR, found in /D0/CMDS, can be specified. So, to the details...

To cater for the situation where the E-PROM has preprogrammed data, the program first asks for the "initial offset (decimal)? " - to start at the beginning, type 0. (BASIC09 will accept hexadecimal numbers as input, if preceeded by the usual '$'). The "names of the modules to add" may use the shortened form if they are in the current directory. After the last one, enter a blank line. Should the last one overflow the E-PROM, one can either program what is presumably the first of several roms, or go onto a subsequent one with the remainder of the routine. For the purposes of Process Control, I need to take control of the Interrupt vectors, so the program allows the option of duplicating these into the E-PROM. Final adjustment of these is done using DEBUG; in response to "any changes? ", answer 'yes'. Finally, a reminder is given to turn on the 25v supply, before the programming proper. After each byte is programmed, it is read back, and if it is not correct, the address of the byte, the one sent, and the one returned are printed.

In addition to the two containing the OS9 routines, my MPU board has space for two other E-PROMs, at $E020 to $EFFF. It certainly has been great to have about ten of the more frequently used utilities in memory at all times - saves a lot of disk wear.

As a Public Service, I will agree to program any E-PROMS (5v 2716 only), but only with material that does not appear to violate any of Microware's, or anybody-else's copyrights- if I think it might, you'll have to get the owners OK, first. Need a 5" 48 tpi floppy, single-sided with the data, enough 2716 roms, the return postage, and $5 beer money for each E-PROM, all done up in a reusable mailing package with a note explaining what you want done.




```
Microware OS-9 Assembler 2.1   09/28/82 22:29:39
B2716 - 2716 PROM BURNER

00001                                  TTL   2716 PROM BURNER
00002                                  NAM   B2716
00003                                  OPT   O
00004   0000 87CD00BF                  MOD   BEND,NAME,$21,$61,ENTRY,0
00005   0000 623233731         NAME    FCS   /b2716/
00006   0004                   INACT   SET   %00110100
00007   003C                   ACT     SET   %00111100
00008   0030                   DDR     SET   %00110000
00009
00010
00011         *******
00012         *
00013         * FIRST ARG (4,S) IS 2716 ADDRESS
00014         * SECOND ARG (8,S) IS DATA
00015         * THIRD ARG (12,S) IS PROM DATA
00016         *  .........all Integers
00017         *
00018         *
00019         * Following is Programmer address:
00020   0012 E000             PPORT   fdb   $E000
00021
00022
00023   0014 AE80FFFA         ENTRY   LDX   PPORT,PCR
00024                *INIT PIA
00025   0018 86FF                     LDA   #$FF
00026   001A C630                     LDB   #DDR
00027   001C E701                     STB   1,X
00028   001E A784                     STA   0,X
00029   0020 E703                     STB   3,X
00030   0022 A702                     STA   2,X
00031   0024 C634                     LDB   #INACT
```

```
         0036 E701                       STB    1,X
         0038 E703                       STB    3,X


         002A 10AE64                     LDY    4,S        GET ARG ADDR
         002D ECA4                       LDD    0,Y        GET ARG
         002F 3404                       PSHS   B          SAVE LOW BITS
         0031 8407                       ANDA   #%00000111 CLEAR HIGHER BITS
         0033 3402                       PSHS   A          SAVE HIGH BITS
         0035 8A80                       ORA    #%10000000 /G TO VIH
         0037 A784                       STA    0,X        TO PIA
         0039 863C                       LDA    #ACT       STROBE
         003B A701                       STA    1,X        INTO
         003D 8634                       LDA    #INACT     LATCH
         003F A701                       STA    1,X
         0041 E784                       STB    0,X        LOW BYTE TO PIA

         0043 10AE6A                     LDY    10,S       GET ADDR OF DATA
         0046 ECA4                       LDD    0,Y        GET DATA
         0048 C1FF                       CMPB   #$FF       BYPASS ALL 1'S
         004A 2716                       BEQ    C
         004C E703                       STB    3,X        DATA TO PIA

         004E 1A10                       ORCC   #%00010000 STOP CLOCK, etc
         0050 108E186A                   LDY    #6250      50 msecs @ 1.00 MHz
         0054 863C                       LDA    #ACT
         0056 A703                       STA    3,X        START PROGRAMMING
         0058 313F              A        LEAY   -1,Y
         005A 26FC                       BNE    A


Microware OS-9 Assembler 2.1   09/28/82 22:32:59
B2716 - 2716 PROM BURNER

         005C 8634                       LDA    #INACT
         005E A703                       STA    3,X        STOP PROGRAMMING
         0060 1CEF                       ANDCC  #%11101111  START CLOCK, etc

         0062 8630              C        LDA    #DDR       SET B-SIDE
         0064 A703                       STA    3,X        TO INPUT
         0066 4F                         CLRA
         0067 A702                       STA    2,X
         0069 8634                       LDA    #INACT
         006B A703                       STA    3,X

         006D 3502                       PULS   A          /G TO VIL
         006F A784                       STA    0,X
         0071 863C                       LDA    #ACT       STROBE
         0073 A701                       STA    1,X        IT
         0075 8634                       LDA    #INACT     IN
         0077 A701                       STA    1,X
         0079 3502                       PULS   A          LOW BITS
         007B A784                       STA    0,X
         007D 12                         NOP
         007E 12                         NOP
         007F 12                         NOP
         0080 12                         NOP
         0081 12                         NOP               TOOL AROUND
         0082 12                         NOP               A BIT

         0083 10AE6C                     LDY    12,S       GET ADDR OF THIRD ARG
         0086 E602                       LDB    2,X        GET DATA
         0088 4F                         CLRA
         0089 EDA4                       STD    ,Y
         008B 39                         RTS
         008C U777CE                     EMOD
         008F                   BCNU     EQU    *
                                         END

00000 error(s)
00000 warning(s)
$008F 00143 program bytes generated
$0000 00000 data bytes allocated
$008B 00187 bytes used for symbols
```

```
PROCEDURE p2716
 0000          DIM i,j,k,l,offset:INTEGER
 0017          BASE 0 \ DIM a(2048):BYTE
 0025          FOR k=0 TO 2047 \a(k)=$FF \NEXT k
 004D          offset=0
 0054          RUN B2716(0,$FFFF,i) \(* TO INITIALISE PORT
 007A
 007B          INPUT "initial offset (decimal)? ",k
 009D
 009E 10       INPUT "name of module to add? ",a$
 00C0          IF a$="" THEN 20
 00CF          OPEN #1,a$:READ
 00DB          WHILE NOT(EOF(#1)) DO
 00E6            GET #1,a(k)
 00F4            k=k+1
 00FF            IF k=2048 THEN
 010C              PRINT "prom too short to hold remainder of"
 0133              PRINT "      ", a$
 013F              PRINT "do you wish to:"
 0152              PRINT "        program this prom, offset $";
 0177              PRINT USING "h4"; offset;
 0184              PRINT "  ...1"
 018E              PRINT " or  read the next prom                  ...2"
 01BC              PRINT " or  quit                        ...anything else"
 01EA              INPUT j
 01EF              IF j=2 THEN
 01FB                offset=offset+$0800
 0207                FOR k=0 TO 2047 \a(k)=$FF \NEXT k
 022F                k=0
 0236                GOTO 15
 023A              ENDIF
 023C              CLOSE #1
 0242              IF j=1 THEN 25
 0251              END
 0253            ENDIF
 0255 15       ENDWHILE
 025C          CLOSE #1
 0262          GOTO 10
 0266
 0267 20       INPUT "vectors? ",a$
 027B          IF a$="yes" OR a$="YES" THEN
 0294            k=$07E0
 029C            FOR i=$FFE0 TO $FFFF
 02AE              a(k)=PEEK(i)
 02BB              k=k+1
 02C6            NEXT i
 02D1          ENDIF
 02D3          GOTO 25
 02D7
 02D8          REM This is a bit of dead code that can
 02FC          REM be activated to dump the E-PROM buffer.
 0328
 0329          j=k-1 \j=0 \k=0
 0342          WHILE j<=l DO
 034F            PRINT USING "x1,h2"; a(j);
 0362            j=j+1
 036D            k=k+1
 037B            IF k>15 THEN
 0384              PRINT
 0386              k=0
 038D            ENDIF
 038F          ENDWHILE
 0393          IF k  0 THEN PRINT  \ ENDIF
 03A2
 03A3 25       INPUT "any changes? ",a$
 03BB          IF a$="yes" OR a$="YES" THEN PAUSE  \ ENDIF
 03D7          INPUT "Vpp ON? ",a$
 03E7
 03E8          l=k-1
 03F3          FOR j=0 TO l
 0404            RUN B2716(j,a(j)+0,i)
 0410            IF i<>a(j) THEN
 0423              PRINT USING "h4,x4,h2,x4,h2",j,a(j),i
 044F            ENDIF
 0451          NEXT j
 045C
 0450
 045E          PRINT CHR$(7); "Vpp OFF?"
```

UNIFLEX Fortran Utility

by Art Matheny
University of South Florida
Biology Dept., LIF 169
Tampa, FL 33620

This paper explains how to construct assembly-language utilities for use in UniFLEX Fortran 77 programs. Two examples are presented— a subroutine and an integer function—and a utility library containing the two subprograms is constructed.

I am an assistant in CAI in the Biology Department of the University of South Florida. We are running the UniFLEX operating system on the Southwest Tech S/09 computer. (UniFLEX is a trademark of Technical Systems Consultants.) Fortran 77 is the version of Fortran which we got from TSC for use with UniFLEX. It came with two other software packages: a relocating assembler and a linking loader. Although this paper is written for those of you who have this software, I hope that it will be of interest to 6809 programmers in general.

One of the nicer aspects of Fortran 77 is the ease with which assembly-language utilities can be added to the language. A library of such utilities can be set up and linked with application program through the use of the "link-edit" command. This paper takes two simple subprograms and describes the steps involved in making them available to Fortran users.

SUBROUTINE EXAMPLE

The first example is a semi-trivial subroutine for swapping two integers. The listing entitled "SWAP" which accompanies this article is a source file for the relocating assembler. The following command is used to assemble it:

++relasmb swap

where "swap" is the filename. The output of the relocating assembler is automatically named "swap.r" where the ".r" suffix indicates that this a relocatable object-code module.

The listing entitled "SWAPTEST" is a Fortran source file to test the subroutine. The key statement is:

```
call swap(i1,i2)
```

which, if all goes according to plan, should exchange the values of i1 and i2. This file should be named "swaptest.f" where the ".f" suffix is required and indicates that this is a Fortran source file. it is compiled by the following command:

```
++f77 swaptest.f +c
```

The "+c" option is important here because it tells the system to compile and assemble the program but not to link it. The linkage step must be done manually. The "f77" command with the "+c" option produces a file named "swaptest.r" which is another relocatable object-code module.

The executable program is produced by the linking loader:

```
++link-edit swaptest.r swap.r +l=F77.runlib +n
+o=swaptest
```

Let us examine this command line in detail:

"link-edit" is the command name for the linking loader. It is described in Chapter 2 of the manual entitled "UniFLEX Linkage Editor".
"swaptest.r" is the module name of the main relocatable object-code module generated by the "f77" command above.
"swap.r" is the module name of the subprogram module generated by the relocating assembler.
"+l=F77.runlib" identifies a library. This argument must ALWAYS be given when linking Fortran programs.
"+n" specifies that the output should be a no-text executable program. One of the three options—"+a", "+n", or "+t"—must always be specified. Refer to Chapter 4 of the "UniFLEX Linkage Editor" manual for an explanation of these three types of executable files.
"+o=swaptest" specifies the name of the executable output file.

The command:

```
++swaptest
```

can now be executed. This program checks to see if the subroutine is functioning properly.

Let us now examine the "SWAP" program in detail. The "name" "global", and "text" statements are apparently necessary. Note that the label "_swap" is composed of the subroutine name itself preceded by an underscore. This is the name that must be used.

The "call swap(i1,i2)" statement in the main program is compiled as follows:

```
ldx #_i2     address of second argument
pshs x
ldx #_i1     address of first argument
pshs x
jsr _swap    subroutine call
leas 4,s     remove arguments from stack
```

Thus the arguments are passed between programs through the stack. However, it is not the argument values themselves which are pushed, but rather the ADDRESSES where those values are to be found. The argument addresses are pushed in reverse order; the first argument is pushed last and ends up on top of the stack. Then the program counter is pushed on top of that by the "jsr" instruction, and at that point the subprogram takes over. Now refering back to the "SWAP" listing, the instruction:

```
_swap leax 2,s
```

merely lets the x-register point to the first argument address in the stack. The stack does not contain the VALUES of i1 and i2, but the ADDRESSES where i1 and i2 are to be found. That is why indirect addressing is used in the operands of the swap section of the subprogram.

Before the subprogram relinquishes control to the main program, it has to do some housekeeping with the stack. The instruction:

```
puls d,y,pc
```

pulls the registers that were pushed earlier and does a return from subroutine. Control is now back in the main program, where the statement,

```
leas 4,s
```

restores the stack pointer by moving it below the argument addresses.

In order to make sure that the subprogram is compatible with the main program, I try to observe the following rules:

(1) No RAM should be used other than the stack.
(2) All registers except the x-register should be restored.

INTEGER FUNCTION EXAMPLE

The second example is a subprogram which is to be used as a function in the Fortran program. The function is called "NACS", which stands for the Number of Actual Characters in a String. You can declare a character string of any size, but there is no end-of-string code. The problem is illustrated by the Fortran program, "nacstest.f", which is listed below. The character string, "line", is declared to be 40 characters long, but the number of characters input by the operator at the "read" statement can be any number 40 or less. Suppose the user inputs the word, "HELLO". Then the string will contain those five letter plus 35 spaces. The system does not insert an end-of-string code. The "nacs" function scans the string in reverse order looking for the first non-blank character. The function value is an integer representing the number of characters up to that point.

The compiler treats functions a little differently from subroutines. it compiles the function reference as follows:

```
leas -2,s    make room in stack for result
ldx #___22     address of the constant 40
pshs x
ldx #_line   address of start of string
pshs x
jsr _nacs
leas 4,s
```

The first instruction moves the stack pointer up to make room for the integer result. Since this is an integer function, and since integers are represented as 2-byte quantities, the compiler leaves two bytes in the stack for the result. It is the responsibility of the assembly-language program to place the result there.

The next four instructions are similar to those of a subroutine call. The addresses of the arguments are pushed in reverse order. It is interesting to note in passing how the compiler handles the second argument, which is the constant, 40. The constant value is stored in memory and its address is given a label—in this case, __Z2. That address is pushed on the stack in the normal manner, and the subprogram can deal with it the same way as it does with variable arguments. Care must be taken not to modify those arguments which might be constants in the main program! There is nothing to prevent you from altering the value of a "constant" in this way, and bugs like that can take weeks to find.

Note that the label, "_nacs", is formed from the function name preceded by an underscore. This is the name that must be used for the assembly-language routine.

Finally, the stack pointer is moved below the arguments in the last statement above. At this point in the main program, the value of NACS is on top of the stack.

The listing entitled "NACS" is a source file for the relocating assembler. Its logic is explained in the program comments and the discussion above. The function result ends up in the y-register, and the statement:

```
nacs2 sty 4,x
```

places the result in the stack at the spot where the main program expects to find it. Note that it is the function value itself, not its address, that is placed there.

The steps for loading "nacstest" are similar to what we did with "swaptest":

```
++f77 nacstest.f +c
++relasmb nacs
++link-edit nacstest.r nacs.r +l=F77.runlib +n
+o=nacstest
```

The output file "nacstest" can now be executed to check the "nacs" function.

LIBRARIES

Utility subprograms can be placed in a library. The Linking Loader will search the library for any external references in the main program. In order to illustrate this, let us form a library with the two subprograms that we now have.

Let's call the library "libxx" and let's place it in the "/lib" directory, which is where all libraries ought to be. The command "lib-gen" is discussed in Chapter 3 of the "UniFLEX Linking Loader" manual. To create a new library with the "swap" subroutine:

```
++lib-gen n=/lib/libxx u=swap.r
```

The argument "n=/lib/libxx" specifies that this is a new library, that it is to be placed in the "/lib" directory, and that it should be named "libxx". The argument "u=swap.r" specifies the module to be updated. The name of the relocatable object-code module is given. The computer should respond:

```
added     _swap      swap.r
Library generation completed.
```

Only the one module, "swap.r" was included in the library, but up to 9 modules could have been added in one step by repeating the "u=" argument.

The "lib-gen" command is also used to update existing libraries. To illustrate this, let's add "nacs.r" to the library:

```
++lib-gen o=/lib/libxx u=nacs.r
```

The computer should respond:

```
copied    _swap      /lib/libxx
added     _nacs      nacs.r
Library generation completed.
```

The "o=/lib/libxx" argument specifies the name of an existing library. The "u=nacs.r" argument specifies the name of the relocatable object-code module to be added.

Other examples of the "lib-gen" command are given in the manual. The name of the library can be changed, and modules can be added, deleted, or updated at will.

To illustrate the use of libraries, let's load "swaptest" again as follows:

```
++link-edit swaptest.r +l=F77.runlib +l=libxx +n
+o=swaptest
```

The "+l=libxx" argument specifies that the Linking Loader should search this library for unresolved external references. In this case the only such reference is "_swap". However as the library grows, there may be dozens of library subprograms called by the main program. Obviously, the use of a library saves a lot of typing as well as mental effort.

```
* * *  NACS  * * *
*
* Number of Actual Characters in String
*
* integer function NACS(LINE,LSIZE)
*
*    LINE = character string
*    LSIZE = declared size of LINE

 name _nacs
 global _nacs
 text
_nacs leax 2,s x points to argument list
 pshs d,y,u save registers

* Set up pointers and counters

 ldd ,x LINE address
 addd [2,x] LSIZE
 tfr d,u u points to LINE character
 ldy [2,x] y is the counter

* Scan LINE in reverse order
* Look for first non-blank character

nacs1 lda ,-u
 cmpa #$20 blank
 bne nacs2
 leay -1,y decrement counter
 bne nacs1
```

```
* Leave result in the stack & return

nacs2 sty 4,x
 puls d,y,u,pc

 end

* * *  NACSTEST  * * *
*
* Test of NACS function
*
* Compile: f77 nacstest.f +c
*
* Externals:
*     nacs.r
*     +l=f77.runlib
*
      external nacs
      integer nacs
      character*40 line
      write(6,1)
    1 format('Input a string: ',$)
      read(5,2) line
    2 format(a40)
      print *,'The string contains ',nacs(line,40),' characters'
      stop
      end


* * *  SWAP  * * *
*
* subroutine SWAP(I1,I2)
*
* I1 and I2 are integers
*
 name _swap
 global _swap
 text
_swap leax 2,s x-reg points to argument list
 pshs d,y save registers

* Do the exchange

 ldd [,x] load I1
 ldy [2,x] load I2
 std [2,x] store I2
 sty [,x] store I1

* Clean up the stack & return

 puls d,y,pc

 end

* * *  SWAPTEST  * * *
*
* Test of SWAP subroutine
*
* Compile: f77 swaptest.f +c
*
* Externals:
*     swap.r
*     +l=f77.runlib
*
      external swap
      integer I1,I2
      write(6,1)
    1 format('What is the first integer? ',$)
      read *,I1
      write(6,2)
    2 format('What is the 2nd integer? ',$)
      read *,I2
      print *,'Before: ',I1,I2
      call swap(I1,I2)
      print *,'After:  ',I1,I2
      stop
      end
```

PL9-MACE

This is a review of the Windrush PL9 compiler and the associated Assembler MACE. It is a small PL compiler. This compiler proper has no built in I/O routines. Windrush supplies a very nice library package that includes I/O, but that is getting a bit ahead of the story. PL9 has some of the nice constructs necessary for structured programming. To quote the manual supplied with it, "PL/9 is intended to be used in applications where assembly language generally predominates, and it has facilities for including subroutines written in assembly language for speed or efficiency."

First of all, let me say that PL9 is very much like a subset of Pascal. A PL9 program contains a number of Procedures, which are, of course, subroutines. You may pass parameters (value or pointer) to your Procedures, and you may declare local variables in addition to the Global ones declared at the beginning of the program. Provision is made for Constants in the form of strings or tables as well as single values. Arrays of one dimension are permitted. The language has the REPEAT UNTIL and the DO WHILE constructs, though there is no counted loop (FOR NEXT).

This implementation allows only INTEGER (16 bit) and BYTE (8 bit) data types. Of course, the BYTE type may be used for Characters as well. It turned out, (though the manual is not very clear on this point) that the last procedure declared becomes the main program. The compiler generates a jump to that point after taking care of the initial statements, which may include an ORIGIN = $XXXX statement, to set the beginning point of the program, and a STACK = $XXXX to set the system stack, which is used for all the variables. The compiler also takes care of including the proper transfer address in the output binary file.

Variable names may be long, up to 127 characters, and ALL are significant. The rules are very standard. The first character must be a letter. Subsequent characters may be numbers, letters (upper case only) or an underline character to separate words in a name, as in VARIABLE_NAME. A variable name preceded by a period (.) causes the compiler to substitute the address of the variable for its value (ie. a pointer). This is of great use in the passing of parameters to procedures.

It is possible to embed assembler code in the program quite easily, and also to declare a variable at an absolute address. The compiler has a built-in Co-

resident Editor, that is very similar to TSC EDIT. To start programming, you simply type "PL9". You then edit your source file. It is possible with editor commands to save the file to disk at any time, or to load a previous file to the editor. The editor contains commands to compile the program and save the compiler binary output to a disk file as well.

The third part of the system, is a debug facility with which you may run the program and insert breakpoints, print variable values, etc.

I found it difficult to get started with the compiler. First of all, it was not obvious that the last procedure is the main program, and that the compiler generates a jump to that procedure. Secondly, there is not a single example program in the manual or on the disk on which the compiler is supplied. The manual is fairly complete, but information is somewhat scattered through its pages, and you remember reading about some statement's syntax, but have to look through the table of contents or thumb through some 44 pages to find it.

My first test of the compiler was to write a simple prime number program to test the code efficiency and speed of execution of the compiled code. The listing here is not a very fancy prime program, but it compiled to a total of about 1500 bytes. The I/O and number conversion library packages comprise more than 1000 bytes of that total. The I/O library routines handle input and output by calling the appropriate routine in FLEX. Execution is quite fast for the algorithm being used. The compiler generates completely position independent code.

An INCLUDE function has been implemented in PL9, which allows you to have text files of standard procedures or library files such as those in the programs here for input and output functions, and include these files in your current program.

The authors of this software obviously had an idea that they should put together an interactive compiler system. The editor, compiler, debugger system all resides in memory at one time. When PL9 is running, you may load a disk file, edit it, save it back to the disk, compile it with a source listing to the terminal, listing of output code to the terminal, binary output file, etc. When the compiler detects an error, it exits back to the editor leaving a pointer (↑) on the CRT pointing at the error and you may edit and recompile immediately. Thus the "interactiveness" of the system. When the compiler has finished, you may execute the code with the debugger if you choose, or simply exit PL9 and run the program.

Four library files in one package, mentioned above, greatly facilitate the use of PL9. IOSUBS is a package for interfacing the terminal through FLEX drivers. It contains several useful routines. NUMCON contains routines that manipulate numbers, converting them from binary to decimal, etc. STRSUBS contains string handling procedures that are the basic ones required for text handling programs, such as editors. FLEX contains routines that interface to FLEX, allowing FILES to be opened and closed, written and read from PL/9 programs.

A second package contains Floating point arithmetic, and a third contains scientific functions. The floating point arithmetic is interesting, because it is done without a FLOAT data type. Floating point values are converted from ASCII or declared as binary constants in a program and stored in an array (4) of byte. All manipulation is done by using pointers at the variables involved. The scientific functions are handled the same way. The following paragraph was of interest to me.

"The accuracy of these routines is limited to four or five significant figures but is usually sufficient for engineering purposes. The algorithm used is one that was published in the November 1980 issue of the '68 Micro Journal as a set of Pascal routines."

Of course the reason for my interest is that I wrote the set of Pascal routines in question. I don't need to test these since I already have done so!

I found a way to get around what might not be a great problem, but certainly an inconvenience in a PL/9 program. The manual for the floating point package says, "To declare a floating point constant or variable in a PL/9 program, a four element BYTE vector must be declared, as follows:"

"BYTE PI $02,$64,$87,$EB;

"PROCEDURE TEST: BYTE FPNUM(4);"

"Once declared, the constant or variable is generally referred to by a pointer (.PI or .FPNUM)."

Though the utilities are supplied to input a number from the terminal, such as 3.1415926 and return the four byte HEX value as above, that is an inconvenience requiring use of utility programs while writing the PL/9 program. Since Ascii to binary conversion routines may be used in the program, I declared the constants as ASCII strings and then used FPASCBIN to convert them to the values for variables used in the calculations.

Though the language is a bit more cumbersome than a higher level language that handles floating point numbers as another data type and would normally interpret algebraic equations, the handling of such values is very regular in the package, and it was quite easy to write the test program. It ran immediately. The addition of file handling capabilities greatly adds to the utility of this package. The string functions include LEN, COPY, COMPARE, CONCATENATE, and STRPOS, which returns the position of a string within another string.

The floating point package contains a few surprise features. In addition to the standard "four functions", it has procedures for square root, copy, compare, float, fix, int, and neg. Of course the conversion procedures FPASCBIN and FPBINASC are included also.

The scientific function package includes the functions LOG, EXP, SIN, COS, TAN, and ATN. I'll have to send Windrush some improvements in a couple of the Scientific functions that I later made. Gee, fellas, I don't mind your using my algorithms, they were published with the intent to make them public domain, but you could have mentioned the name of the author of the original package!

Seriously, though, this is a rather complete package which might very well find use in a "dedicated control system" where a bit of number crunching must be done. It almost seems to me that someone out there has been listening to my rantings in my column for a couple of years, and decided to implement an efficient "intermediate level" language.

The companion Assembler is called MACE for (M)6809 (A)ssembler and (C)o-resident (E)ditor. It also has the interactive feature of the resident Editor. Windrush has done us a favor by using the same Editor in both the compiler and the Assembler. The commands for invoking the Assembler are exactly the same as those for the Compiler. This Assembler has some rather nice features. It allows what are called "local labels". These are valid from one "regular" label to the next, and duplicate local labels may be used in several places in a program. The Assembler also supports conditional assembly. It is very fast, and it is very nearly compatible with the TSC 6809 assembler, and CRASM8.

If you write programs for dedicated control applications and want something in a higher level language a bit above

Assembler, but with the interactive capability of BASIC, you will find this compiler to your liking. I found it to work completely as indicated in the manual.

Regarding the programs listed here, the first, the Prime number program uses the I/O library and the Number conversion library. PRINT (.MESSAGE) calls the PRINT library routine with a pointer to the ascii text MESSAGE. The routine is like PSTRNG in FLEX. It prints characters until it finds a null rather than the $4 required by PSTRING. INPUT, requires a pointer to a buffer, and the length of the buffer to be passed to it. It will input information from the terminal until the buffer is full or a CR is detected. It terminates the string with a null. PRDEC prints an integer value as an unsigned number. You might be interested in how I formatted my numeric output. The list of primes prints in neat columns on an 80 column terminal.

The second program is a test of the conversion routine for ASCII to binary and the reverse, and it tests FPDIV, the divide routine, and SIN, the sine routine, printing the result to the terminal. Floating point numbers must be converted to strings before being output to the terminal, and from strings on input from the terminal.

Conclusions

This is a well thought out package. Performance in terms of ease of use and lack of bugs was impressive. The documentation is adequate, but would profit from a bit of organization, and the inclusion of a few demonstration programs. The inclusion of the library I/O procedures for communication with a terminal will help anyone who had bought the compiler to write some test programs and get a feel for it before launching off into the world of ROM based stand alone control systems.

This language is somewhat lower level than Pascal or BASIC, but it is most usable, and will, because of its nearness to Assembler level, probably result in very efficient programs in terms of bytes and execution time.

Flash! The assembler MACE is also available in a cross assembler version to run on the 6809 but assemble 6800 code.

Prices

MACE/XMACE Combined $98.00 USA
PL9 w/libraries and MATH PKG $198.00 USA

Additional information may be secured from:

Windrush Micro Systems, Ltd.
Worstead Walsham, Norfolk NR28 9SA
England

Ron Anderson - - -

```
/* PROGRAM TO FIND PRIME NUMBERS IN PL9 */

ORIGIN = 0;

STACK = $8000;

GLOBAL
   INTEGER
      MAX,
      CANDIDATE,
      TESTNUM;
   BYTE
      PRIME; /* BOOLEAN VARIABLE */

INTEGER PRIME1 1;  /* THE FIRST THREE PRIMES */
INTEGER PRIME2 2;
INTEGER PRIME3 3;

/* PROCEDURE TO OUTPUT AN INTEGER TO TERMINAL */

ASMPROC OUTNUM (INTEGER);
   GEN $AE,$62;          /*LDX 2,S GET POINTER TO INTEGER 0/
   GEN $C6,$FF;          /* SET FLAG TO OUTPUT LEADING SPACES */
   GEN $7E,$CD,$39;      /* JMP OUTDEC (HIDDEN RTS) 0/

/* PROCEDURE TO INPUT AN INTEGER FROM COMMAND LINE 0/

ASMPROC INNUM (INTEGER);
   GEN $BD,$CD,$4B;      /* JSR INDEC */
   GEN $AF,$F8,$02;      /* STX [2,S] 0/
   GEN $39;              /* RTS 0/

PROCEDURE PRIMES;

   OUTNUM (.PRIME1);  /* OUTPUT THE FIRST THREE PRIMES */
   OUTNUM (.PRIME2);
   OUTNUM (.PRIME3);

   INNUM (.MAX); /* GET MAX NUMBER */

   CANDIDATE = 5;

   WHILE CANDIDATE <= MAX
   BEGIN
      PRIME = TRUE;
      TESTNUM = 3;

      WHILE TESTNUM * TESTNUM < CANDIDATE AND PRIME
      BEGIN
         TESTNUM = TESTNUM + 2;
         IF (CANDIDATE / TESTNUM) * TESTNUM = CANDIDATE
            THEN PRIME = FALSE;
      END;

      IF PRIME THEN OUTNUM (.CANDIDATE);
      CANDIDATE = CANDIDATE + 2;
   END;
   JUMP $CD03;
ENDPROC;

/* LISTING 1, TEST PROGRAM FOR TERMINAL I/O PROCEDURES 0/


ORIGIN=$8000;
STACK=$8000;

GLOBAL BYTE NUMBER, LIMIT;

/* STRINGS DEFINED BELOW 0/

BYTE MESSAGE $D,$A,'INPUT LIMIT ',$D,$A,4;
BYTE LFCR $D,$A,$4;


/* OUTPUT A CHARACTER TO THE TERMINAL 0/

PROCEDURE PUTCHR (BYTE CHAR);
   ACCA = CHAR; /* SPECIAL INSTRUCTION TRANSFERS CHAR TO ACCUM. A */
   CALL $CD1B;
ENDPROC;


/* PRINT A STRING.  POINTER TO START, TERMINATED BY $4 */

PROCEDURE PSTRNG (BYTE .STRING) ;
          INTEGER POINTER; BYTE CHAR;
```

```
   POINTER = .STRING;
   REPEAT
      CHAR = MEM(POINTER);
      PUTCHR (CHAR);
      POINTER = POINTER+1;
   UNTIL CHAR = 4;
ENDPROC;


/* GET A CHARACTER FROM THE TERMINAL */

PROCEDURE GETCHR (BYTE .CHAR);
   CALL $CD15;
   [CHAR]=ACCA;
ENDPROC;


/* START OF MAIN PROGRAM FOR TESTING ABOVE PROCEDURES */

PROCEDURE MAIN;

   NUMBER = 1;
   PSTRNG (.MESSAGE);
   GETCHR (.LIMIT);
   PSTRNG (.LFCR);

   WHILE NUMBER <= (LIMIT - $30)
   BEGIN
      PUTCHR (NUMBER+$30);
      PUTCHR ($20);
      NUMBER = NUMBER + 1;
   END;
   JUMP $CD03;

ENDPROC;
```

OSM Assembler Review

by E. M. (Bud) Pass, Ph.D.
Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers, GA 30207
Telephone Number 404-483-1717/457E

GENERAL INFORMATION

OSM is a macro assembler designed to assemble programs written for the 6809. It comes in two versions, one to run under FLEX and one to run under OS/9. Either version will assemble programs for either operating system's program protocol and file format.

In addition to the basic ability to recognize the 6809 assembly language and to process symbolic address expressions, it has the ability to support structured assembly, the ability to define macros with parameters, the ability to define conditional assembly directives, and the ability to assemble programs stored in modular form.

The discussion of OSM below is intended to familiarize the reader with the concepts of OSM and to introduce the novice to several advanced assembler language concepts, applicable also to other assemblers. Finally, there is a brief comparison of OSM with the TSC 6809 Macro Assembler and the Microware 6809 Interactive Assembler.

CALLING AND CONTROLLING OSM

The command lines used to execute OSM are as follows:

FLEX:

```
OSM <source file spec>
    [<object file spec>]
    [<switches>]
    [<parameters>]
```

OS/9:

```
OSM <source path spec>
    [O=<object path spec>]
    [O=<switches>]
    [P=<parameters>]
```

When running OSM under FLEX, the source file specification defaults to the extension of ".TXT", and the object file specification defaults to the extension of ".BIN" and the file name of the source file.

When running OSM under OS/9, the standard input and output files are used, and the object code file will default to the execution directory, using the same file name as the source file; it will have its public read and execute, and owner read, write, and execute permission bits turned on.

Under either operating system, only one source file name is allowed, and one object file name is allowed, but the assembler directives "LIB" or "USE" may be used within a source file to specify multiple input files.

There are several option switches, as alluded to above, which assist the user in controlling OSM's actions, from the command line. They are as follows:

```
Y...(FLEX ONLY) Automatic object code file deletion.  (OFF)
L...No list option turned on (no assembler output).  (LIS)
S...No Symbol Table list turned on (suppressed).  (SYM)
G...Single line list on FCC, FCB, FDB turned on.  (GEN)
B...(FLEX ONLY) No Object Code saved on disk.  (BIN)
P[number]...PAG option turned on, and optional starting page
     number.  (NOP)
N...Use line numbers.  (OFF)
C...Set to position independent mode.  (OFF)
T...Start printing comment lines in address field.  (OFF)
```

There are several other option switches which assist the user in controlling OSM's actions from the OPT directive, which appears in the source file, not on the command line.

ASSEMBLER DESCRIPTION

OSM accepts standard Motorola assembler free-format source language statements. The assembler language statement may contain ASCII characters with hex values between $20 and $7E, inclusive.

Four fields on the source line are recognized by Motorola-compatible assemblers. These consist of the LABEL, the OPERATOR, the OPERAND, and the COMMENT fields, separated by one or more space characters. The format just described may be represented as follows:

```
[label] [operator] [operand] [comment]
```

If the label field is present and starts with an asterisk, the line is assumed to be a comment line only.

The restrictions and options for each field are as follows:

LABEL FIELD

1. The label must not be preceded by spaces.
2. The label must be unique.
3. The label must begin with a letter, and lower case letters are converted to upper case.
4. The label must consist of the characters 'A' - 'Z', 'a' - 'z', '0' - '9', '$', '.', '_'.
5. Only the label's first few characters are significant, unless the SYM directive is used to change the length from a default of 6 for FLEX or 7 for OS/9 to a minimum of 3 and a maximum of 30.
6. The label must be followed by a space.

OPERATOR FIELD

1. The operator field follows most of the rules for the label field (except for starting in the first column).
2. Operators which require a register specifier may delete or include the separating space (for 6800 compatibility).

OPERAND FIELD

1. Operands are considered as composed of one or more expressions.
2. Operands are evaluated for the expression value and the addressing mode used.
3. In case an instruction requires no operand, the operand field is considered part of the comment field.

COMMENT FIELD

1. This field is optional and is composed of the remainder of the characters on the line after the operator and operand fields.
2. A total line length of 128 characters including the carriage return is allowed.

EXPRESSIONS

Expressions consist of combinations of symbols, numbers, and program/data counters separated by operators. Arithmetic is done in 16 bit integer precision. Expressions must not contain spaces, and the expression is terminated when a space, carriage return, or unexpected character is found.

Following is a summary of the formats for numbers:

BASE	PREFIX	CHARACTERS
Decimal	none	0-9
Binary	%	0-1
Octal	@	0-7
HexaDecimal	$	0-9,A-F,(a-f)
ASCII	'	$20-$7E

Following is a summary of the operators:

operation	operator	form
add	+	value+value
subtract	-	value-value
multiply	*	value*value
divide	/	value/value
and	&	value&value
or	|	value|value
not	!	!value
shift right	>>	value>>count
shift left	<<	value<<count
equal	=	value=value
less than	<	value<value
greater than	>	value>value
less or equal	<=	value<=value
greater or equal	>=	value>=value
not equal	<>	value<>value

Symbols are labels, as described above. The current program counter

value is represented by the character '*'. The current storage counter is defined by the character '.' in OS/9 mode and the character '*' in FLEX mode.

AUTOMATIC LABEL GENERATION

OSM is capable of automatically generating label of the form "Lxxxxx" where 'x' is a decimal digit (0-9). An automatically-generated label is represented by a colon, followed by an optional macro label offset number. This capability has greatest power when used within macros, since it simplifies the definition of macros containing labels.

An example is the macro equivalent to the BASIC statement PRINT. A macro would be created as follows:

```
PRINT   MACRO
        LDX #:1
        JSR PSTRNG
        BRA :2
:       FCC "&1",4
:       EQU *
        ENDM
```

It could be called as follows:

```
        PRINT "HI"
```

The code generated would be as follows:

```
        LDX #L00001
        JSR PSTRNG
        BRA L00002
L00001  FCC "HI",4
L00002  EQU *
```

OBJECT CODE FORMAT

OSM can generate object code files using one of two selectable formats. The first is the default FLEX binary file format. The second is the OS/9 module format.

A FLEX binary file is composed of records formatted as follows:

```
Byte 0    Start of record ($02, the ASCII STX)
Byte 1    Most significant byte of load address
Byte 2    Least significant byte of load address
Byte 3    Number of data bytes following
Byte 4-n  Binary object code data to be loaded
```

A FLEX binary file may also contain an optional transfer address record, used to indicate the starting address for the program. A FLEX transfer address record is formatted as follows:

```
Byte 0  Transfer address indicator ($16, ASCII ACK)
Byte 1  Most significant byte of the transfer address
Byte 2  Least significant byte of the transfer address
```

The OS/9 or 'no record format' is turned on by the option 'NOR' or

when an OS/9 module is started with a MOD statement. The object code output under this mode uses no record format except that imposed by the optional OS/9 MOD and ENDD statements. Object code is assumed to be one continuous memory block in this format..

MACROS

Macro capabilities, even of the elementary form found in OSM, can assist the programmer in structuring assembler code and can save much coding and maintenance time, especially when used in coordination with conditional assembly (described later). Macros differ from subroutine calls in that the source language lines represented by the macro are inserted into the program every time the macro is called, rather than being placed at one location within the program and being branched to for each call.

The first line of a macro definition must have an operator field containing 'MACRO' and a nonblank label field, which is used as the name of the macro being defined. The source code text lines following are placed into a reserved area of memory referenced later by the macro name. The last line of the macro definition must have an operator field containing 'ENDM'.

To call the text referenced by a macro name, the macro name is used as if it were an operator. If any parameters are to be passed to the macro, then the macro name is followed by a space and one or more expressions seperated by commas.

While a macro is being expanded, parameters are substituted symbolically wherever they appear. Parameters are represented within a macro by the character '&' followed by a non-zero digit. If an undefined parameter is referenced, it is assumed to be null.

Macros may call other macros directly or from within other macros. Parameters are referenced on a local level to each macro. That is if a macro is called from within another macro, the current macro's parameters are pushed on to a stack. The macro calling line may use the parameter substitution characters to pass on a parameter to a subsequent macro.

If 'EXITM' appears in the operator field of a line being processed within a macro, the expansion of that macro is terminated. It is normally used with conditional assembly.

Assembler calling line parameters are referenced by the '&' character followed by 'A' to 'I' for parameters 1 to 9 of the assembler calling line parameter buffer. These are global in nature, and may be referenced inside or outside of a macro.

CONDITIONAL ASSEMBLY

Assemblies may be structured with using conditional assembly directives. A very convenient use of conditional assembly capabilities is for customizing programs to a set of parameters, since the parameters may often be used to control the assembly. Conditional assembly differs from 'IF' statements in BASIC and in other languages in that the conditional is evaluated at assembly time, not at run time, and the source language statements in the ranks of the

conditional assembly are either included or excluded from the assembly at that point. In BASIC, the statements are always included in the program, but are dynamically executed or skipped depending upon run-time conditions.

'IF' will cause the lines up to the next matching 'ENDIF' or 'ELSE' to be included only if the expression is not equal to zero 'IFN' will include the lines only if the expression is zero. In either case, if a matching 'ELSE' is found before the matching 'ENDIF' the test condition is reversed for the succeeding lines.

The syntax of these directives is as follows:

```
IF <expression>   or
IFN <expression>
```

'IFC' and 'IFNC' will do a string comparison rather than expression comparison. These forms require a logical string comparison expression.

The syntax of these directives is as follows:

```
IFC <expression>   or
IFNC <expression>
```

'IFP1' and 'IFP2' allow for assembly of the included lines only on pass 1 or on pass 2. They require no expressions.

MODULAR SOURCE LANGUAGE

The LIB and USE directives may be used to introduce additional source language files into a file being assembled. They are very useful for large programs or for subroutine libraries, and provide many of the capabilities at the source language level normally provided by a link editor at the object language level. They assist in the development of structured, modular assembly source language programs. LIB and USE have the same meaning and are simply alternative names for the text inclusion function.

The syntax of these directives is as follows:

```
LIB <file name>   or
USE <file name>
```

COMPARISON WITH OTHER ASSEMBLERS

OSM has certain features which are similar to the TSC and Microware assemblers, and others which are different from either of them. The primary similarities among all three assemblers are related to the use of the Motorola format and Motorola 6809 mnemonics. Often when a software product such as an assembler does not support one feature or concept, it has another to compensate for the missing feature or concept.

The Microware assembler does not support the 6800 compatible operators such as LDAA and INX, macro definitions, the expanded FCC statement, the IF, IFN, IFC, IFNC, ENDIF directives, the shift operators '<<' and '>>', command line parameters, and symbols containing '_', as do both

of the TSC and OSM assemblers.

The TSC assembler does not support separate program and data counters, the MOD and ENDD statements, the OS9 statement, the PCS statement, the IFP1 and ENDC directives, and symbols containing '$' or '.', as do both of the Microware and OSM assemblers.

Only the Microware assembler supports the IFEQ, IFNE, IFLT, IFGT, IFLE, IFGE directives.

Only the TSC assembler supports symbols with lower case letters as being unique from the correspondiog symbols with upper case letters.

Only the OSM assembler supports automatic label generation and the generation of FLEX formatted binary files under OS/9 and vice versa.

The OPT directives are different among the three assemblers.

CONCLUSIONS

Which is the better assembler among the three primary assemblers for the 6809 FLEX-OS/9 market, Microware, OSM, or TSC? Of course, each has its stronger and its weaker points. The weighting factors for comparison of the three assemblers would very significantly depending upon the applications and preferences present in a particular environment. There is no single best answer. All three assemblers work well in their areas.

If cross-generation of binary files is very important in an application, OSM is the only choice. This appears to be the only critical, non-substitutable feature among the three assemblers which is not shared by at least one other of the assemblers. Automatic label generation, lower-case recognition, and other features unique to only one of the three assemblers are substitutable, but cross-generation is not.

OSM does not have the experience nor time in field enjoyed by the TSC and Microware assemblers, simply because it is so new. This does not imply that OSM has more bugs or problems than the other two assemblers. All three assemblers have known bugs, inconsistencies, and peculiarities. It does imply that OSM does not yet have the name recognition of the other two assemblers, a problem which it may overcome, given enough time.

OSM is a product of Lloyd I/O and is marketed by Frank Hogg Laboratory, 130 Midtown Plaza, Syracuse, NY 13210 (315) 474-7856.

BIT BUCKET

HOYT STEARNS ELECTRONICS
4131 E. Cannon Dr.
Phoenix, Az. 85028
602 996 1717
October 24,1982

Don Williams, Publisher
Computer Publishing Center
68 Micro Journal
5900 Cassandra Smith
Box 849
Hixson, Tn 37343

Dear Mr. Williams,

Here is an interesting set of programs for the Color Computer to make use of the "page 1" mode for a 64K machine.

It is important to be aware of the difference between MAP type 1 and PAGE 1: MAP type 1 replaces the ROM space from $8000 to $FEFF with the upper half of the 64K ram, thus the ROM data is no longer accessible. PAGE 1 mode merely causes the $8000 to address the upper 32K of ram instead of the lower 32K: the ROMS remain accessible. Note that the display address is always from RAM and independent of the mode or page selection.

The difficulty in using PAGE 1 mode is that the entire useable RAM is switched - so the program that is doing the switching and the stack are both instantaneously lost. The way around this is to somehow put a small set of routines into page 1 that are essentially duplicates of the ones in page 0 at the same addresses. These are "boundary crossing routines".

The only way to put these routines into page 1 without having at least a small number of bytes in some common memory is to use MAP type 1 (the mode used by FLEX). The upper 32K in MAP type 1 is the same physical memory as the lower 32K in Page 1 mode, so if the boundary crossing programs are copied up in MAP type 1 such that their addresses will be the same in PAGE 1 memory, then the programs will be duplicated in both pages.

There are several uses for PAGE 1:

1. Any program can be disassembled even if it occupies the same space as the disassembler itself.
2. Many more graphics Pages are available.
3. There may be two different resident programs that can be multiplexed.

Here is an example in using these routines to disassemble a program:

1. Modify an existing Basic disassembler by finding all PEEKs used for examining the target program and changing them to USR0. At the beginning of the disassembler, define USR0 as &H7F00.
2. LOADM the above set of routines and execute &H7F65 to construct the boundary crossing links
3. Load the program to be disassembled, normally.
4. Execute &H7FA3 to move it over to P1
5. Load the modified disassembler normally, and execute it.

Here are the listings:

```
        NAM P1MOVE8
        ORG $7EEC
TPUT2   STA <$FFD5   P0 TEMPORARY

        ORG $7EF8
TGET2   STA <$FFD5  SET PAGE 1

        ORG $7EFE
TLAST   RTS         END OF TEMP. P0 CODE

        ORG $7F63

*CONSTRUCT BOUNDARY CROSSING CODE AT THE TOP OF BOTH PAGES.

MPLACE  LDD #$50FF
        EXG CC,A    DISABLE INTERRUPTS, SAVE MASKS
        EXG DP,B    SET DIRECT PAGE TO $FF
        PSHS U,Y,X,B,A SAVE REGISTERS

        LEAX PUT2,PCR POINT TO  COMMON ROUTINES
        LEAY $7F00,X  SET TO COPY TO TOP OF MAP1 RAM
        STA $FFDF   SET MAP TYPE 1
        LDB #LAST-PUT2  THE SIZE OF THE ROUTINES

LOOP1   LDA B,X   DO THE COPYING UP
        STA B,Y
        DECB
        BPL LOOP1

        STA $FFDE   SET MAP TYPE 0 AGAIN

*NOW COPY THE ROUTINES UP $100 BYTES IN PAGE 1 TO PUT THEM AT
THE TOP.  THE TOP OF P1 IS NOT ACCESSIBLE IN MAP TYPE 1 SINCE
THE SAM USES THIS SPACE.  THIS CODE USES THE DUPLICATED CODE TO
MOVE ITSELF UP!

        LDA #9   COUNTER
        PSHS A   ON STACK

        LEAX TPUT2,PCR  POINT TO EXISTING TEMPORARY CODE

LOOP2   LBSR TGET2  PULL 2 BYTES IN FROM P1 ,X
        LEAX $100,X POINT $100 BYTES HIGHER
        LBSR TPUT2  PUT THEM BACK IN P1 HIGHER
        LEAX 2-$100,X  ADJUST X BACK TO TEMPS
        DEC ,S      DECREMENT THE COUNTER
        BPL LOOP2   UNTIL DONE

        LEAS 1,S    DUMP THE COUNTER
        PULS CC,DP,X,Y,U,PC RESTORE THE REGISTERS AND RETURN
```

*THIS ROUTINE COPIES ALL OF P0 TO P1 SO A PROGRAM LOADED NORMALLY CAN BE MOVED OVER.

```
P1COPY LDD #$50FF
       EXG CC,A   DISABLE INTERRUPTS PRESERVING MASKS
       EXG DP,B   SET DP PRESERVING OLD DP
       PSHS U,Y,X,B,A SAVE EVERYTHING

       LEAX PUT2,PCR  TOP OF AREA TO COPY
LOOP3  LDD -2,X  GET TWO BYTE IN P0
       BSR PUT2  PUT THEM IN P1
       CMPX #0
       BNE LOOP3 UNTIL DONE

       PULS CC,DP,X,Y,U,PC RESTORE AND RETURN
```

* THESE ROUTINES SIMULATE PEEK AND POKE. ONLY THEY ACT FROM P0 TO P1.

```
P1PEEK PSHS U,X,Y SAVE SOME REGISTERS
       JSR $B3ED   DO BASIC'S INTCNV
       TFR D,X   PUT ADDRESS IN X
       LDD #$50FF
       EXG CC,A   DISABLE INTERRUPTS
       EXG DP,B   SET DP
       PSHS B,A   AND SAVE THE OLD ONES
       LEAX -1,X  ONLY WANT 1 BYTE, NOT 2
       BSR GET2   GET 2 BYTES FROM P1
       CLRA       CLEAR OUT UNWANTED BYTE
       PULS CC,DP RESTORE CC,DP
       JSR $B4F4  RETURN VALUE TO BASIC'S USR(1
       PULS X,Y,U,PC RESTORE AND RETURN
```

* THE POKE ROUTINE HAS NOT BEEN TESTED. IT USES A STRING VARIABLE IN A USER FUNCTION WHERE THE FIRST TWO BYTES ARE THE ADDRESS AND THE 3RD IS THE DATA.

```
P1POKE PSHS U,Y,X
       LDD #$50FF
       EXG CC,A   DISABLE INTERRUPTS
       EXG DP,B
       PSHS B,A
       LDX 2,X   GET POINTER TO STRING
       LDB 2,X   GET BYTE DATA
       LDX ,X    GET ADDRESS FROM STRING
       LEAX 1,X  ADJUST FOR AUTO DECREMENT IN PUT1
       BSR PUT1  PUT A BYTE IN P1
       PULS CC,DP,X,Y,U,PC

* HERE ARE THE ROUTINES THAT ARE IN BOTH PAGES

PUT2   STA <$FFD5   * SET P1

* THE INSTRUCTIONS WITH * IN THE COMMENTS FIELD NEED TO BE IN
P0, THE OTHERS IN P1 ONLY.

       STB ,--X   STORE IN P1
       BRA EXIT

PUT1   STA <$FFD5   * SET P1
       STB ,-X   STORE IN P1
       BRA EXIT

GET2   STA <$FFD5   * SET P1
       LDB ,X++   GET DATA FROM P1

EXIT   STA <$FFD4  SET P0
LAST   RTS         *

       END  EMPLACE
```

TELECON SYSTEMS
1155 Meridian Ave., Suite 218
San Jose, CA 95125
(408) 275-1639

UNIVERSAL CROSS REFERENCE REPORT GENERATOR

TELECON SYSTEMS announces the Universal Cross Reference Report Generator. UXREF uses an external file, in clear text, which specifies the rules for generating the report.

The external file is customer configurable to meet the requirements of any language: Assembler, Basic, C, English, Pascal, etc.

External files are provided for Basic and C. The package is available for immediate delivery and is priced in the $50.00 range.

Dear Don:

Enclosed you will find two programs for submittal to 68 Micro Journal.

The first one is called "MOD.CMD", and is a modem communications program and is written for a 6809 cpu using Flex (tm) with 6850 ACIAs for both the modem port and terminal inport. The outport may be whatsoever, because the program uses the output routine supplied by user. The program transmits and receives all printable ASCII-chars (.TXT files). The size of the the files are only limited by the user memory in receive mode. The program uses interrupts both from the terminal and from the modem, because that eliminates handshaking problems in high speed transmissions and gives the user full control of the program. Also it is not necessary to switch between Rx and Tx. That is why both ACIAs have to be wired for IRQ.

At startup the IRQ-vector is saved and at exit (cntrl F) it is reloaded.

The second program, "HEX.CMD", simply echoes in ASCII everything to the screen whatever is typed on the keyboard. For example <Return> is echoed as ^0D, line feed as ^0A, space as 20,... The benefit of the program is is that if you have customized several keys, it is possible to see the value of the keys.

Please feel free to use these programs in your magazine, that also has a great number of readers in Sweden.

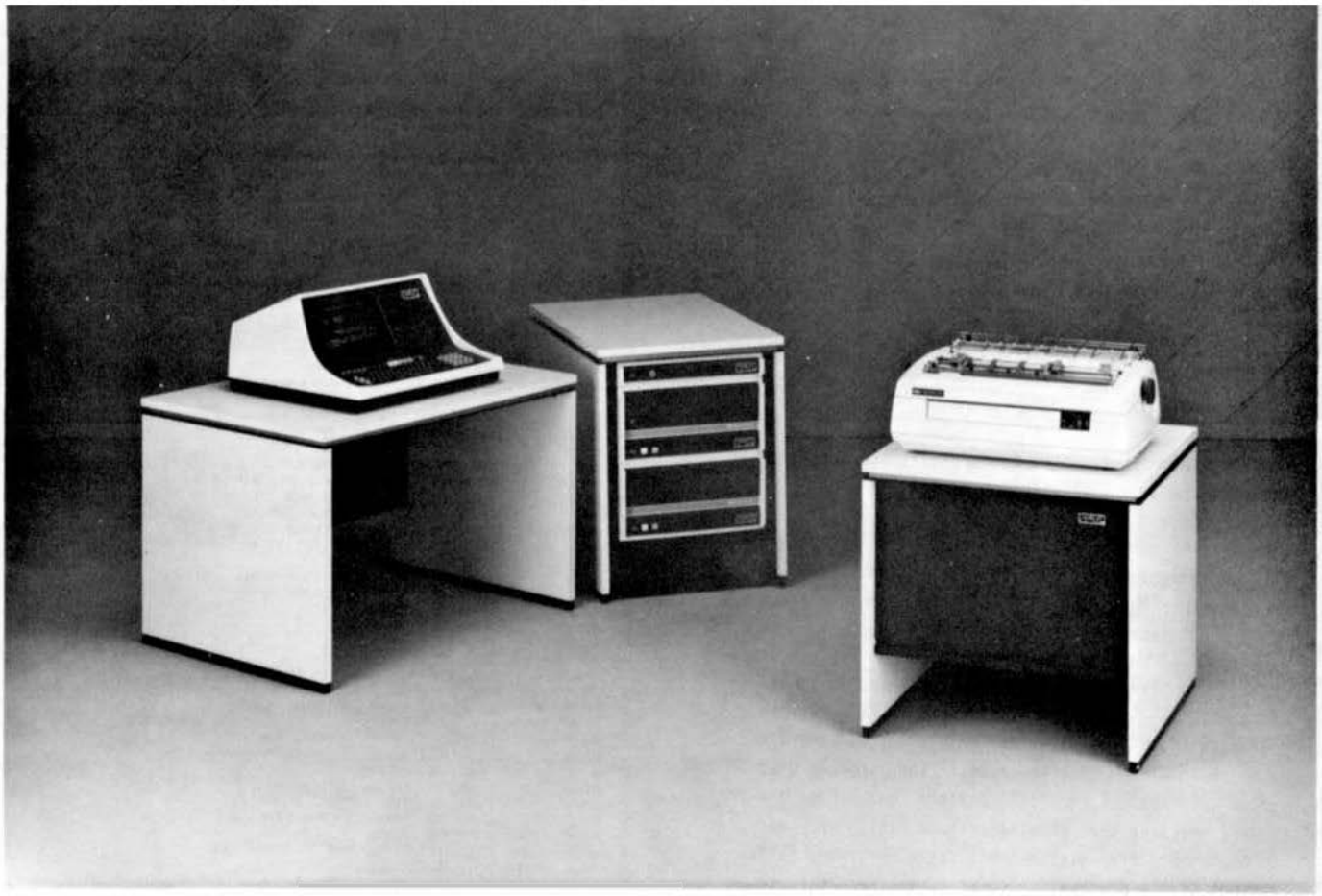
Yours truly

Esko Antikainen (SM5AKP) and Avo Kask (SM0KVO)

Arvodesv. 17
S-126 46 HAGERSTEN
SWEDEN

Hoderv 8
S-184 02 OSTERSKAR
SWEDEN


```
                        *
                        *       MODEMPROGRAM    for ACIa 6850, CPU 6809.
                        *       and     FLEX dos
                        *
                        *       Both ACIA's have to be wired for IRQ !
                        *
                        *       Written by Esko Antikainen ( SM5AKP )
                        *                  and Avo Kask ( SM0KVO )
                        *
                        *       JUNE    1987
                        *
                        *
                        *       SYNTAX: MOD.CMD
                        *
                        *       SYSTEM  EQUATES
C003    WARMS   EQU     $C003
C01E    PSTRNG  EQU     $C01E
E000    MODEA   EQU     $E000   MODEMPORT (ACIA)
E004    TERMID  EQU     $E004   TERMINALPORT (ACIA)
DFCD    IRQVEC  EQU     $DFCD   ADDRESS TO IRQ ROUTINE
C018    INBUFF  EQU     $C018
C033    SETEXT  EQU     $C033
C02D    GETFIL  EQU     $C02D
9A03    FMSCLS  EQU     $9A03
0446    FMS     EQU     $0446
CD40    FCB     EQU     $CD40
C03F    RPTERR  EQU     $C03F
C024    PCRLF   EQU     $C024
CC29    HEXOUT  EQU     $CC29
                *
0000 20  10     START   BRA     STAR10
0002 02                 FCB     2       VERSION NUMBER
                *
                *       SYSTEM  CONSTANTS ( adapt to your system )
0003 05                 ECHO    FCB     5       ^E ECHO ON/OFF
0004 14                 TRANSM  FCB     $14     ^T SEND  DISKFILE
0005 13                 SAVE    FCB     $13     ^S SAVE TEXT ON RECEIVE
0006 17                 WRITE   FCB     $17     ^W WRITE RECEIVED TEXT TO DISK
0007 06                 RETUR   FCB     $06     ^F RETURN TO FLEX
0008 1B                 ESCAPE  FCB     $1B     ^[ ESCAPE CHARACTER
0009                    IRQSAV  RMB     2       SAVE SYSTEM IRQVECTOR
000B 00                 EFLG    FCB     0       ECHO ON=FF, OFF=00
000C #20C               PRINTB  FDB     BUFSTA
000E FF                 SAVFLG  FCB     $FF     0=SAVE RECEIVED TEXT-
000F 7E  C01B           TBOUT   JMP     $C01B   FLEX OUTPUT ROUTINE
0012 E005               ACIACE  FDB     TCNTRL+1
0014 E004               ACIACR  FDB     TERMIR
0016 E009               MODACE  FDB     MODCR+1
0018 E008               MODCR   FDB     MODEA
                *
001A DE  DFCD   START0  LDU     IRQVEC
001C 9F  09             STX     IRQSAV  SAVE SYSTEM IRQVECTOR
001E DE  01EC           LDU     #ENTER
0020 DF  DFCD           STU     IRQVEC  INTERRUPTROUTINE
0022 B6  03             LDA     #3      ACIA MASTER RESET
0024 A7  9F 0014        STA     [ACIACR]
0027 A7  9F 0018        STA     [MODCR]
002A B6  95     START1  LDA     #%10010101 ENABLE INTERRUPT
002C A7  9F 0014        STA     [ACIACR] SET UP BOTH ACIAS
002F A7  9F 0018        STA     [MODCR]
0032 A6  9F 0012        LDA     [ACIACE] Dummy READ RESET IRQ-FLAG
0035 A6  9F 0016        LDA     [MODACE] *
0038 DE  #210           LDU     #STATXT DISPLAY MENU
003B BD  CD1E           JSR     PSTRNG
003E 30  CDC4           JSR     PCRLF
0041 3C  EF     LOOP    CWAI    #%11101111 ENABLE IRQ
0043 20  FC             BRA     LOOP
0045 DE  01EA   DISFIL  LDU     #DISFIL TEXT FOR SETTING DISKFILE
0047 30  CD1E           JSR     PSTRNG
004A C6  15             LDB     #$15
004C E7  9F 0014        STB     [ACIACR] NO IRQ FROM KEYBOARD
004F 8D  CD1D           JSR     INBUFF  INPUT FILENAME
0051 8D  CD24           JSR     PCRLF
0053 C6  95             LDB     #%10010101
0055 E7  9F 0014        STB     [ACIACR] IRQ AGAIN
0058 DE  C040           LDU     #FCB
005A 8D  CD2D           JSR     GETFIL
005C 1025 6142          LBCS    ERROR
0060 86  01             LDA     #1      EXT=TXT
0062 8D  CD33           JSR     SETEXT
0064 B6  01             LDA     #1
0066 A7  84             STA     ,X
0068 8D  0446           JSR     FMS     OPEN READ
006A 1026 0132          LBNE    ERROR
006E 8D  0446   RELOOP  JSR     FMS     GET CHR
0070 1026 0128          LBNE    ERROR
0074 4D  05             TSTA    EFLG
0076 27  0C             BEQ     LOP
0078 91  0F             CMPA    TBOUT   OUTPUT TO TERMINAL
007A 81  00             CMPA    #00
007C 26  06             BNE     LOP
007E B6  09             LDA     #09
0080 9D  0F             JSR     TBOUT   OUTPUT A LINE FEED
0082 B6  00             LDA     #00
0084 E6  9F 0018 LOP    LDB     [MODACR]
0087 C4  02             ANDB    #%00000010
0089 27  F8             BEQ     LOP     NOT READY YET
008B A7  9F 0016        STA     [MODACE]
008E 20  BB             BRA     RELOOP
0090 0E  07F5   WRITE   LDU     #BUFSTA WRITE TEXT TO DISK
0093 B6  CD1E           JSR     PSTRNG
0096 C6  15             LDB     #$15
0098 E7  9F 0014        STB     [ACIACR] NOT TERMINAL IRQ
009B 8D  CD1D           JSR     INBUFF
009D C6  95             LDB     #%10010101
009F E7  9F 0014        STB     [ACIACR] TERMINAL IRQ AGAIN
00A2 DE  DFCD           LDU     IRQVEC
00A4 8D  CD2D           JSR     GETFIL
00A6 1025 00EC          LBCS    ERROR
00AA B6  03             LDA     #3
00AC 8D  CD33           JSR     SETEXT  EXT=.TXT
00AE B6  02             LDA     #2
00B0 B7  84             STA     ,X
00B2 8D  0446           JSR     FMS     OPEN FOR WRITE
00B4 1026 00DE          LBNE    ERROR
00B8 108E 020C          LDY     #BUFSTA
```



THE COMPLETE BUSINESS SYSTEM

+Multiuser+Highly Expandable+Cost Effective

S+ THE CONCEPT

The S+ system is a modular computer system in which all portions of the hardware and software are designed to work together in the most efficient way possible. An S+ single user system with floppy disk storage is a competitive and cost effective entry level system. Unlike most other small computers being sold as "personal", or "small business" machines, the S+ system may be expanded to maximum capabilities using this same hardware and software. You cannot end up with a DEAD END system that cannot be expanded and whose software is not compatible with larger machines. A basic S+ system may be expanded to thirty-two users, a megabyte of main memory and hundreds of megabytes of hard disk storage by simply plugging in, or connecting the desired upgrade equipment.

TOTAL DESIGN—Hardware and Software

The S+ system is an integrated hardware and software design. The two complement and enhance each other in this system. The UniFLEX® operating system used in the S+ systems is patterned after the Bell Laboratories UNIX® operating system, one of the most admired and widely used operating systems in the world. Instead of being an afterthought, the software is part of the design of the S+ system. You can be sure that with this approach that all parts of the computer operate with maximum efficiency and cost effectiveness.

THE CENTRAL PROCESSOR

The basic S+ system is configured with 256K bytes of memory and can be expanded to more than 1 million bytes. An efficient and fast hardware memory management system is used to allocate the available memory among the users on a dynamic basis. As little as 8K bytes, or the entire memory—if needed—can be used by any individual user. This makes it possible to run very large programs on the system, but it also uses no more memory than necessary for a particular job. The increase in cost effectiveness of this system over crude and outdated bank switching arrangements is dramatic.

The central processor runs in both user and supervisor states. It can detect and reject a defective user program. It is impossible for a user program to go bad and stop the entire system, as can happen quite easily in less sophisticated systems.

Task switching is accomplished by use of a multiple map RAM memory, with sixty-four individual task maps. Each task can access from 4 to 64 K-bytes of memory. Multiple tasks may be used in programs that require more than 64K bytes of memory for execution. When a task is completed the memory is automatically released for other use.

SOFTWARE

The S+ operating system, UniFLEX® is a multiuser, multitasking operating system based on the UNIX® operating system that has been used for many years on Digital Equipment Corp. PDP-11 series minicomputers. It is considered one of the most sophisticated and "user friendly" operating systems available. Variations of UNIX® are rapidly becoming standard on mini and larger microcomputers.

A large variety of languages are available for use with the system. These include FORTRAN, COBOL, BASIC, and Pascal. Word processing packages are also available to give you full text processing capability on the system.

Applications programs are available in large quantities in many fields. This includes general business, medical, dental, veterinary, library and real estate management; plus others. Since the system is multiuser it can also be connected to cash registers to produce a point-ofsale terminal system combined with the computer. The possibilities for application of this system are endless.

THE I/O SYSTEM

The S+ system is totally interrupt driven. All terminal and printer I/O devices connect to an I/O bus separate from the main bus. Up to thirty-two separate devices may be connected to the I/O bus at any one time. If I/O activity is great enough to cause an unacceptable slowdown in system operation, a separate I/O processor can be installed in the system. This plug-in option removes all I/O handling overhead from the main processor and allows operation of up to thirty-two external devices at 9,600 baud. Without an integrated total design, as in the S+ system, it would become impractical to use a UNIX®type operating system in a situation with heavy terminal I/O activity.

DISK STORAGE

A wide range of disk storage capacity is available for the S+ system, from 2.5 M-byte floppy disks to an 80 M-byte Winchester and many sizes between. All disk controllers use direct memory access (DMA) type operations to maximize data transfer and to minimize overhead on the main processor. The Winchester disks also use intelligent controllers along with DMA transfers to preserve the performance that these type devices are capable of giving. Without this distributed intelligence the system performance would be greatly degraded. The UniFLEX®operating system is designed to work at maximum efficiency with this type disk system. The data transfer rates achieved by this combination rival those of large minicomputers.

COMMUNICATIONS

A high speed local network communications system is available to interconnect S+ systems. The VIABUS® network will allow communication between systems at data rates of over 400K baud. Such a system makes it possible to share data between local systems in an efficient and low-cost manner.

AVAILABLE SOON

Tape backup—20M-Byte in less than 15 minutes on a standard ¼ inch cartridge.

Mini-Wini—5 and 10 M-Byte Winchesters—5¼ inch package. Winchester performance, for smaller systems in a small package. UniFLEX® compatible design.

Large Capacity—190 and 340 M-Byte Winchesters, plus SMD cartridge drives.

UniFLEX is a registered trademark of Technical Systems Consultants, Inc.

UNIX is a registered trademark of Bell Labs.

VIABUS is a registered trademark of Southwest Technical Products Corporation.

```
00D8 109C 0C     WLOP   CMPY  POINTER
00DB 27 0B              BEQ   WSTOP
00DD A6 A0              LDA   ,Y+
00DF BD D406            JSR   FMS        WRITE TO DISK
00E2 1026 00CC          LBNE  ERROR
00E6 20 F0              BRA   WLOP
00E8 86 04       WSTOP  LDA   #4
00EA A7 84              STA   ,X
00EC BD D406            JSR   FMS        CLOSE FILE
00EF 1026 00BF          LBNE  ERROR
00F3 8E 020C            LDX   #BUFSTA
00F6 9F 0C              STX   POINTER    CLEAR BUFFER
00F8 0E 2F              JMP   START1

00FA 86 03       EXIT   LDA   #3         MASTER RESET ACIA
00FC A7 9F 0014         STA   [ACIACR1]
0100 86 15              LDA   #$15       CONFIGURE NORMAL KEYBOARD ACIA
0102 A7 9F 0010         STA   [ACIACR]
0106 9E 09              LDX   IRQSAV
0108 BF DFC0            STX   IRQVEC     RESET SYSTEM IRQ
010B 7E CD03            JMP   WARMS
010E BD D403     ESCA   JSR   FMSCLS     CLOSE OPEN FILES
0111 86 FF              LDA   #$FF
0113 97 0E              STA   SAVFLG     DO NOT SAVE TEXT ON RECEIVE
0115 8E 020C            LDX   #BUFSTA
0118 9F 0C              STX   POINTER    RESET TEXTPOINTER
011A 0E 2F              JMP   START1     JUMP TO NORMAL RECEIVE ROUTINE
011C E6 9F 0010  INTEP  LDB   [MODCR1]
0120 2A 2A              BPL   CHRIN
0122 A6 9F 0016         LDA   [MODARE1]
0126 84 7F              ANDA  #$7F       STRIP PARITY
0128 9D 0F              JSR   TEROUT
012A 0D 0E              TST   SAVFLG
012C 26 56              BNE   RTI
012E 9E 0C              LDX   POINTER
0130 8C CC20            CMPX  MEMEND
0133 27 0D              BEQ   WRISTO     MEMORY FULL
0135 A7 80              STA   ,X+
0137 9F 0C              STX   POINTER
0139 81 0D              CMPA  #$D
013B 26 DF              BNE   INTEP
013D 86 0A              LDA   #$A
013F 9D 0F              JSR   TEROUT     CRLF TO TERMINAL
0141 3B                 RTI
0142 8E 01D6     WRISTO LDX   #FULTXT    MEMORY FULL
0145 BD CD1E            JSR   PSTRNG
0148 8E 002F            LDX   #START1    NEW RETURNADDRESS
014B AF 6A              STX   10,S
014D 3B                 RTI
014C E6 9F 0010  CHRIN  LDB   [ACIACR1]  CHARACTER INPUT HANDLER
0152 2A 2A              BPL   RTI
0154 A6 9F 0012         LDA   [ACIADR1]  GET CHAR
0158 84 7F              ANDA  #$7F       STRIP PARITY
015A 81 07              CMPA  RETUR      RETURN TO FLEX ?
015C 27 27              BEQ   RET
015E 81 05              CMPA  ECHO       ECHO ON/OFF ?
0160 27 20              BEQ   EFOSET
0162 81 05              CMPA  SAVE       SAVE TEXT ?
0164 27 29              BEQ   SAVSET
0166 81 01              CMPA  TRANSM     TRANSMIT TEXT ?
0168 27 36              BEQ   TRANIT
016A 81 06              CMPA  WRITE      WRITE TO DISK ?
016C 27 38              BEQ   DISWRI
016E 81 00              CMPA  ESCAPE     START ALL OVER AGAIN ?
0170 27 3A              BEQ   ESCAP
0172 0D 00              TST   EFLG       ECHO ??
0174 27 0C              BEQ   OUT        IF SO ECHO
0176 9D 0F              JSR   TEROUT     CHARACTER TO TERMINAL
017A E6 9F 0010  OUT    LDB   [MODCR1]
017E C4 02              ANDB  #%00000010
0180 27 F8              BEQ   OUT
0182 A7 9F 0016         STA   [MODARE1]
0186 3B          RTI    RTI
0187 8E 00FA     RET    LDX   #EXIT
018A AF 6A              STX   10,S       NEW RETURNADDRESS
018C 3B                 RTI
018D 03 00       EFOSET COM   EFLG
018F 20 F5              BRA   RTI
0191 0F 0E       SAVSET CLR   SAVFLG
0193 8E 0290            LDX   #SAVTXT
0196 BD CD1E            JSR   PSTRNG
0199 BD CD24            JSR   PCRLF
019C 8E 020C            LDX   #BUFSTA
019F 9F 0C              STX   POINTER    CLEAR THE BUFFER
01A1 3B                 RTI
01A2 8E 004E     TRANIT LDX   #DISFIL
01A5 AF 6A              STX   10,S       NEW RETURNADDRESS
01A7 3B                 RTI
01A8 8E 00A5     DISWRI LDX   #WRIDIS
01AB AF 6A              STX   10,S       NEW RETURNADDRESS
01AD 3B                 RTI
01AE 8E 010E     ESCAP  LDX   #ESCA
01B1 AF 6A              STX   10,S       NEW RETURNADDRESS
01B3 3B                 RTI

01B4 A6 01       ERROR  LDA   1,X        ERROR BYTE
01B6 81 08              CMPA  #8
01B8 26 16              BNE   STOP
01BA 86 04              LDA   #4
01BC A7 84              STA   ,X
01BE BD D406            JSR   FMS        CLOSE FILE
01C1 A6 01              LDA   1,X
01C3 26 08              BNE   STOP
01C5 8E 0207            LDX   #REDIIT
01C8 BD CD1E            JSR   PSTRNG
01CB BD CD24            JSR   PCRLF
01CC 0E 2F              JMP   START1
01CE BD CD3F     STOP   JSR   RPTERR
01D1 BD D403            JSR   FMSCLS
01D4 0E 2F              JMP   START1
01D6 07 0D 0A 4D FULTXT FCC   7,/Memory full !!/,4
01DA 6F 72 79 20
01DE 66 75 6C 6C
01E2 20 21 21 04
01E6 07 4E 61 6D DISTXT FCC   7,/Name of diskfile to transfer ? /,4
01EA 65 20 6F 66
01EE 20 64 69 73
01F2 6B 66 69 6C
01F6 65 20 74 6F
01FA 20 74 72 61
01FE 6E 73 66 65
0202 72 20 3F 20
0206 04
0207 07 54 72 61 REDIIT FCC   7,/Transmission ready/,4
020B 6E 73 6D 69
020F 73 73 69 6F
0213 6E 20 72 65
0217 61 64 79 04
021B 5E 54 3D 74 STATXT FCC   /^T=transmit file,^S=save text on receive,^W=write to disk /,$A,$D
021F 72 61 6E 73
0223 6D 69 74 20
0227 66 69 6C 65
022B 2C 5E 53 3D
022F 73 61 76 65
0233 20 74 65 78
0237 74 20 6F 6E
023B 20 72 65 63
023F 65 69 76 65
0243 2C 5E 57 3D
0247 77 72 69 74
024B 65 20 74 6F
024F 20 64 69 73
0253 6B 20 0A 0D
0257 5E 45 3D 65        FCC   /^E=echo on,off,  ^F=return to FLEX,      ^[=escape/,$A,$D
025B 63 68 6F 20
025F 6F 6E 2C 6F
0263 66 66 2C 20
0267 20 5E 46 3D
026B 72 65 74 75
026F 72 6E 20 74
0273 6F 20 46 4C
0277 45 58 2C 20
027B 20 20 20 20
027F 20 5E 5B 3D
0283 65 73 63 61
0287 70 65 0A 0D
028B 4D 4F 44 45        FCC   /MODEM ON LINE/,$A,$D,4
028F 4D 20 4F 4E
0293 20 4C 49 4E
0297 45 0A 0D 04
029B 07 53 61 76 SAVTXT FCC   7,/Saving text into buffer /,4
029F 69 6E 67 20
02A3 74 65 78 74
02A7 20 69 6E 74
02AB 6F 20 62 75
02AF 66 66 65 72
02B3 20 04
02B5 07 57 72 69 WRITXT FCC   7,/Write saved text to disk. FILENAME ? /,4
02B9 74 65 20 73
02BD 61 76 65 64
02C1 20 74 65 78
02C5 74 20 74 6F
02C9 20 64 69 73
02CD 6B 2E 20 46
02D1 49 4C 45 4E
02D5 41 4D 45 20
02D9 3F 20 04
02DC             BUFSTA RMB   1          START OF BUFFER
                        END   START

0 ERROR(S) DETECTED

SYMBOL TABLE:

ACIADR 0010  ACIARE 0012  BUFSTA 020C  CHRIN  014C  DISFIL 004E
DISTXT 01E6  DISWRI 01A8  ECHO   0005  EFLG   0000  EFOSET 018D
ERROR  01B4  ESCA   010E  ESCAP  01AE  ESCAPE 0000  EXIT   00FA
FCB    C840  FMS    D406  FMSCLS D403  FULTXT 01D6  GETFIL CD2D
INBUFF CD1B  INTEP  011C  IRQSAV 0009  IRQVEC DFC0  LOOP   0044
LOP    0002  MEMEND CC20  MODARE 0016  MODCR  0010  MODEM  E000
OUT    017A  PCRLF  CD24  POINTE 000C  PSTRNG CD1E  REDIIT 0207
RELOOP 000B  RET    0185  RETUR  0007  RPTERR CD3F  RTI    01A1
SAVE   0005  SAVFLG 000E  SAVSET 018F  SAVTXT 029B  SETEIT CD33
START  0000  START0 001A  START1 002F  STATXT 021B  STOP   01CE
TERMIN E004  TEROUT 000F  TRANIT 01A2  TRANSM 0001  WARMS  CD03
WLOP   00D0  WRIDIS 00A5  WRISTO 0142  WRITE  0006  WRITXT 02B5
WSTOP  00E8

                        * PRINTS NEW CHARS FROM KEYBOARD TO DISPLAY

                        * WRITTEN BY ESKO ARTILAINEN ( SFEMAP )
                        * AND AVO RASK ( SMEVD ) IN JUNE 1982

                        * FLEX DOS EQUATES

               CD1B  INCHNP  EQU  $CD1B
               CD24  PCRLF   EQU  $CD24
               CD1E  PSTRNG  EQU  $CD1E
               CD03  WARMS   EQU  $CD03
               CD3C  OUTHEX  EQU  $CD3C

                        * MONITOR EQUATE
```

```
                FDC9  INCH   EQU   $FDC9
                
C100                         ORG   $C100
                
C100 86  06     START0 LDA   #5      ERASE SCREEN
C102 BD  CD1B          JSR   PUTCHR
C105 BD  CD24          JSR   PCRLF
C108 8E  C15F          LDX   #MESS1
C10B BD  CD1E          JSR   PSTRNG
C10E BD  CD24     START2 JSR   PCRLF
C111 C6  0C            LDB   #12     ANOUNT OF CHARS ON EACH ROW
C113 BD  FDC9     START  JSR   INCH   INCH IN MONITOR
C116 84  7F            ANDA  #$7F    STRIP PARITY
C118 81  7F            CMPA  #$7F    EDIT TO FLEX
C11A 26  04            BNE   OUT
C11C 1027 0BE3         LBEQ  WARMS
C120 B7  C15E     OUT  STA   BYTE
C123 8E  C15E          LDX   #BYTE
C126 BD  CD3C          JSR   OUTHEX  OUTPUT BYTE IN HEX
C129 86  3B            LDA   #';
C12B BD  CD1B          JSR   PUTCHR
C12E 96  C15E          LDA   BYTE
C131 80  20            SUBA  #$20    IS IT CONTROL CHAR ?
C133 25  0E            BCS   CNTRL   BRANCH IF YES
C135 96  C15E          LDA   BYTE
C138 BD  CD1B          JSR   PUTCHR
C13B 5A                DECB
C13C 27  C9            BEQ   START2  12 CHARS DONE YET ?
C13E 20  D0            BRA   START
C140 86  5E       CNTRL LDA   #'^     OUTPUT AN UP-ARROW
C142 BD  CD1B          JSR   PUTCHR  FOR CNTRL CHARS
C145 96  C15E          LDA   BYTE    GET CHAR
C147 8B  40            ADDA  #$40    MAKE IT AN ASCII CHAR
C149 BD  CD1B          JSR   PUTCHR
C14C BD  C150          JSR   SPACE   OUTPUT A SPACE
C14F 5A                DECB
C150 27  B6            BEQ   START2  POW FULL ON DISPLAY ?
C152 20  09            BRA   START
C154 86  20       SPACE LDA   #$20
C156 BD  CD1B          JSR   PUTCHR
C159 39                RTS
C15A           BYTE   RMB   1
C15B 41 33 65 0F MESS1 FCC   /Ascii - Hex - Keyboard converter/,$0D,$A,4
     C163 69 20 2D 20
     C167 48 65 78 20
     C16B 2D 20 41 0D 65
     C170 79 62 6F 61
     C175 72 64 20 63
     C179 6F 6E 76 65
     C17D 72 74 65 72
     C181 0D 0A 04
                
                EXB    START0

0 ERROR(S) DETECTED
```


Don Williams
'68' Micro Journal
5900 Cassandra Smith
Computer Publishing Center
PO Box 849
Hixson TN 37343

Rushmore Micro Systems
622 E. Tallent
Rapid City, SD 57701
(605) 342-8797

August 3, 1982

Dear Don,

The accompanying program is the converse of one submitted by Bud Pass several months ago. It converts FLEX binary files to S1 format. This conversion would be useful to provide tape output for programs stored in FLEX format on disk.

The output is normaly to the terminal although it may be redirected using the FLEX "O" command. An "S9" record will terminate the file unless the "N" option is used.

Sincerly,

Byron Everett
Rushmore Micro Systems

```
                        OPT   PAG
                ********************************************************
                *                                                      *
                *  - FLEX BINARY to S1 - S9 FORMAT                     *
                *    CONVERSION PROGRAM                                *
                *                                                      *
                *    Written by B. Everett                             *
                *    Placed in the Public Domain, 1982                 *
                *                                                      *
                *    Calling sequence :                                *
                *      S1S9 [file-spec] I+N]                           *
                *    The "N" option causes the omission of the S9      *
                *                                                      *
                ********************************************************
                *                                                      *
                ********************************************************
                *
                * EQUATES
                
CB4B  SYSFCB  EQU   $CB4B
CC14  LINPTR  EQU   $CC14     FLEX LINE POINTER
CD03  WARMS   EQU   $CD03
CD1B  PUTCHR  EQU   $CD1B
CD1E  PSTRNG  EQU   $CD1E     PCRLF then PDATA
CD24  PCRLF   EQU   $CD24
CD2D  GETFIL  EQU   $CD2D
CD33  SETEXT  EQU   $CD33     SET EXTENSION
CD3F  RPTERR  EQU   $CD3F     REPORT ERROR
D406  FMS     EQU   $D406
                *
C100                ORG   $C100
C100           CKSUM RMB   1        CHECKSUM
C101           ADDR  RMB   2
C103 0000      YTEMP FDB   0

C105 8E  CB4B  START LDX   #SYSFCB
C108 BD  CD2D        JSR   GETFIL
C10B 25  69          BCS   ERROR

C10D 86  00          LDA   #0       Default extension = BIN
C10F BD  CD33        JSR   SETEXT
C112 86  01          LDA   #1
C114 A7  84          STA   ,X
C116 BD  D406        JSR   FMS      OPEN INPUT FILE
C119 26  5B          BNE   ERROR

C11B 86  FF          LDA   #$FF
C11D A7  88 3B       STA   59,X     SET BINARY FILE
                *
                * Loop for each record in the Flex binary file
                *
C120 BD  59     LOOP1 BSR   GETB
C122 25  35           BCS   EOFR

C124 81  02           CMPA  #2       START OF BIN RECORD ?
C126 26  F8           BNE   LOOP1    NO, LOOP

C128 BD  66           BSR   GET16
C12A 10BF C101        STY   ADDR     SAVE ADDRESS
C12E 17  0089         LBSR  GETBYT   GET LENGTH
C131 B7  C104         STA   YTEMP+1
C134 10BE C103        LDY   YTEMP
                *
                * Loop for each output record required
                *
C138 10BC 0000  LOOP2 CMPY  #0
C13C 25  E2           BLS   LOOP1

C13E CC  0010         LDD   #16      GET DEFAULT RECORD LENGTH
C141 108C 0010        CMPY  #16
C145 24  02           BHS   L2

C147 1F  20           TFR   Y,D      FIX RECORD LENGTH IF REQ.
C149 34  04     L2    PSHS  B        SAVE LENGTH
C14B BD  52           BSR   OUTREC   OUTPUT 1 LINE
C14D 35  04           PULS  B
C14F BE  C101         LDX   ADDR
C152 30  85           LEAX  B,X      UPDATE ADDR
C154 BF  C101         STX   ADDR
C157 20  DF           BRA   LOOP2

C159 BE  CC14  EOFR   LDX   LINPTR
C15C A6  84           LDA   ,X
C15E 81  2B           CMPA  #'+
C160 27  0C           BEQ   OPTION

C162 8E  C1F2  PS9    LDX   #S9M
C165 BD  CD1E         JSR   PSTRNG
C168 BD  CD24  EXIT   JSR   PCRLF
C16B 7E  CD03         JMP   WARMS
C16E A6  01    OPTION LDA   1,X
C170 81  4E           CMPA  #'N      "N" OPTION ?
C172 26  EE           BNE   PS9      NO

C174 20  F2           BRA   EXIT
```

```
C176 BD    CD3F   ERROR    JSR    RPTERR     REPORT ERROR
C179 20    ED              BRA    EXIT

C17B 8E    C840   GETB     LDX    #SYSFCB
C17E BD    D406            JSR    FMS
C181 27    0C              BEQ    GETBX      NO ERROR, EXIT

C183 34    04              PSHS   B
C185 E6    01              LDB    1,X
C187 C1    08              CMPB   #8         EOF ?
C189 26    EB              BNE    ERROR      NO, REPORT ERROR

C18B 1A    01              SEC               SET CARY
C18D 35    84              PULS   B,PC

C18F 39           GETBX    RTS

C190 8D    E9     GET16    BSR    GETB
C192 25    0A              BCS    GET16X     EXIT ON EXCEPTION

C194 34    02              PSHS   A
C196 8D    E3              BSR    GETB
C198 1F    89              TFR    A,B        MOVE TO Dlower
C19A 35    02              PULS   A          GET Dupper
C19C 1F    02              TFR    D,Y        Y <- ADDR
C19E 39           GET16X   RTS
                  *
                  * - OUTREC -
                  *   OUTPUT 1 LINE IN S1 FORMAT
                  *
                  * ASSERT 0 < D <= 16 : requirement of call
C19F 7F    C100   OUTREC   CLR    CKSUM
C1A2 8E    C1EF            LDX    #S1M
C1A5 BD    CD1E            JSR    PSTRNG     PRINT "S1"
C1A8 1F    98              TFR    B,A        A <- LENGTH
C1AA 8B    03              ADDA   #3         ADJUST IT
C1AC 8D    1A              BSR    PUTBYTE
C1AE B6    C101            LDA    ADDR
C1B1 8D    15              BSR    PUTBYT
C1B3 B6    C102            LDA    ADDR+1
C1B6 8D    10              BSR    PUTBYT
C1B8 8D    30     OLOOP1   BSR    GETBYT
C1BA 8D    0C              BSR    PUTBYT
C1BC 31    3F              LEAY   -1,Y       DEC LENGTH
C1BE 5A                    DECB              DEC COUNT
C1BF 26    F7              BNE    OLOOP1

C1C1 B6    C100            LDA    CKSUM
C1C4 43                    COMA
C1C5 8D    01              BSR    PUTBYT     OUTPUT CHECKSUM
C1C7 39                    RTS

C1C8 34    02     PUTBYT   PSHS   A
C1CA 46                    RORA
C1CB 46                    RORA
C1CC 46                    RORA
C1CD 46                    RORA
C1CE 8D    0D              BSR    PUTDIG
C1D0 A6    E4              LDA    ,S         GET LOWER NIBBLE
C1D2 8D    09              BSR    PUTDIG
C1D4 35    02              PULS   A
C1D6 BB    C100            ADDA   CKSUM
C1D9 B7    C100            STA    CKSUM      UPDATE CHECKSUM
C1DC 39                    RTS

C1DD 84    0F     PUTDIG   ANDA   #$F
C1DF 8B    30              ADDA   #'0
C1E1 81    39              CMPA   #'9
C1E3 2F    02              BLE    PUTDI

C1E5 8B    07              ADDA   #'A-'9-1
C1E7 7E    CD18   PUTDI    JMP    PUTCHR

C1EA 8D    8F     GETBYT   BSR    GETB
C1EC 25    88              BCS    ERROR

C1EE 39                    RTS

C1EF 53 31 04     S1M      FCC    'S1',4
C1F2 53 39 04     S9M      FCC    'S9',4
                           END    START
```

0 ERROR(S) DETECTED

SYMBOL TABLE:

ADDR	C101	CKSUM	C100	EOFA	C159	ERROR	C176	EXIT	C168
FMS	D406	GET16	C190	GET16X	C19E	GETB	C17B	GETBX	C18F
GETBYT	C1EA	GETFIL	C02D	L2	C149	LINPTR	CC14	LOOP1	C13F
LOOP2	C138	OLOOP1	C1B8	OPTION	C16E	OUTREC	C19F	PCRLF	C024
PS9	C162	PSTRNG	CD1E	PUTBYT	C1C8	PUTCHR	CD18	PUTDI	C1E7
PUTDIG	C1DD	RPTERR	CD3F	S1M	C1EF	S9M	C1F2	SETEXT	CD33
START	C105	SYSFCB	C840	WARMS	CD03	XTEMP	C103		

DON,

HERE IS A NEAT MEMORY EXAMINE & CHANGE PROGRAM WRITTEN FOR PUBLIC DOMAIN BY ONE OF OUR PROGRAMMERS, RICHARD PLADSEN.

ALSO- I WANT TO BRING TO YOUR ATTENTION A PROBLEM THAT WE FOUND WITH AN SS-30 PROTOTYPING BOARD MANUFACTURED BY SOUTH EASTERN MICRO. THE +8 VOLT UNREGULATED SUPPLY PINS ARE OFF BY 1 & WILL PUT +8 VOLTS UNREGULATED ON THE R/W LINE. THIS BOARD WAS BEING OFFERED REAL CHEAP, BUT THEY DON'T MENTION THIS PROBLEM.
I USE A DISK CONTROLLER BOARD FOR MY FLOPPY MADE BY SOUTH EASTERN MICRO. IT SEEMS TO WORK OK @ 2MHZ IF I STRETCH MEMORY BY 500NS.
WE USE AN IEEE-488 INTERFACE HERE FOR INSTRUMENTATION & ARE WONDERING IF THIS WOULD BE WORTH WHILE SELLING AS AN SS-30 BOARD.
I WOULD LIKE TO MENTION A PATCH FOR FLEX'S DEBUGGER. WHENEVER ONE ATTEMPTS TO WRITE TO A WRITE-ONLY LOCATION THE DEBUGGER REPLIES WITH A "WHAT?" & JUMPS OUT OF THE EXAMINE/CHANGE MODE. SOMETIMES THIS CAN BE IRRITATING & TSC WAS OF NO HELP.
THE FOLLOWING PATCH WILL WORK IF YOUR VERSION IS THE SAME AS OURS:

```
ORG $5F97
RPT 6
NOP
END
```

ROBERT ZEFF
ZEFF ADVANCED PRODUCTS
(209) 577-4268

```
          ******************************************************
          *        [illegible] - A DYNAMIC, RELOCATABLE,        *
          * SCREEN ORIENTED MEMORY EXAMINE/CHANGE UTILITY      *
          *    --FOR THE FLEX DISK OPERATING SYSTEM--          *
          *  Written by: Richard Pladsen                        *
          *               Zeff Advanced Products Co.            *
          *               2135 [illegible] Ave.                 *
          *               Modesto, CA   95350                   *
          *               9-22-82 (209) 577-4268                *
          ******************************************************

          *FLEX EQUATES
  D3F9    VOUTCH  EQU  $D3F9   indirect address vector, output char to terminal.
  D3E5    [illegible] EQU  $D3E5   indirect address vector, input char [illegible].
  CD03    [illegible] EQU  $CD03   exit address.
  000A    LF      EQU  10      linefeed.
  000D    CR      EQU  13      carriage return.
  0008    BS      EQU  8       backspace.
  000B    VT      EQU  11      cursor up.
  000C    FF      EQU  12      cursor down.
          *
          INIARG  MACRO
          [illegible] SET  0
                  ENDM
          *
          DEFARG  MACRO  [illegible]   allocation macro.
          &1      SET  ARGS
          ARGS    SET  ARGS+&2
                  ENDM
          *
 C100             ORG  $C100
          OVERLAYABLE INITIALIZATION.
 C100             INIARG
```

```
         C100                     DEFARG LINEBUF,5 assign variable names to stack offsets.
         C100                     DEFARG VAL,7
         C100                     DEFARG MODE,1
         C100                     DEFARG HCOUNT,1
         C100                     DEFARG ACOUNT,1
         C100 20    01    MEMEX1  BRA    START
         C102 02          VER     FCB    2          relocatable version.
         C103 32    76    START   LEAS   -ARGS,S
         C105 1F    43            TFR    S,U        all variables are offsets of U.

                          *CLEAR THE SCREEN AND BUILD A 16 BIT ADDRESS.
         C107 17    01F6  MEMEX1  LBSR   CLRSCR     clear the screen.
         C10A CC    0010          LDD    #$0010
         C10D 17    01E2          LBSR   LOADCUR
         C110 30    8D 0291       LEAX   ADRPRM,PCR print address prompt.
         C114 17    01FD          LBSR   PSTRNG
         C117 CC    0022          LDD    #$0022
         C11A 17    01D5          LBSR   LOADCUR
         C11D 5F                  CLRB              accb keeps track of # digits typed in (4 max).
         C11E 30    C4            LEAX   LINEBUF,U  point X to input-editing buffer area on stack.
         C120 17    01EA  INADR   LBSR   INCH       get a character and process it.
         C123 81    0D            CMPA   #CR        is it a carriage return?
         C125 27    2D            BEQ    RETU
         C127 81    08            CMPA   #BS        is it a backspace?
         C129 27    11            BEQ    BACKUP
         C12B 17    0234          LBSR   CLASS      classify character.
         C12E 2A    F0            BPL    INADR      N-bit is set if hexadecimal.
         C130 C1    04            CMPB   #4         only accept 4 chars (+cr) maximum.
         C132 27    EC            BEQ    INADR
         C134 A7    85            STA    B,X        store char in buffer.
         C136 17    01B0          LBSR   OUTCH      echo it.
         C139 5C                  INCB              increment the char count.
         C13A 20    E4            BRA    INADR
         C13C 5D          BACKUP  TSTB              backspace during address entry.
         C13D 27    E1            BEQ    INADR      don't backspace if nothing was typed.
         C13F 5A                  DECB              decrement the char count.
         C140 34    10            PSHS   X
         C142 30    8D 0339       LEAX   BACKSP,PCR
         C146 17    01CD          LBSR   PSTRNG
         C149 35    10            PULS   X
         C14B 20    D3            BRA    INADR
         C14D A7    85    RETU    STA    B,X        put CR in buffer and process the number.
                          the ascii address is now built.
         C14F 17    01C9          LBSR   BCVHEX     returns a right justified in D if B is 0, a points to ascii string.
         C152 ED    45            STD    VAL,U
         C154 6F    47            CLR    MODE,U     default to hex entry mode.

                          *DUMP BLOCK ON SCREEN
         C156 17    016C  DUMPS   LBSR   PORTADR
         C159 E6    46            LDB    VAL+1,U    dump only whole pages of memory.
         C15B 34    04            PSHS   B
         C15D 6F    46            CLR    VAL+1,U    clear the lsb of address.
         C15F CC    0300          LDD    #$0300
         C162 17    0180          LBSR   LOADCUR
         C165 30    8D 02A6       LEAX   DUMPH,PCR  print the header.
         C169 17    01AA          LBSR   PSTRNG
         C16C C6    10            LDB    #16        do 16 rows.
         C16E E7    49            STB    ACOUNT,U
         C170 C6    10    HLOOP1  LDB    #16        do 16 columns of hex first.
         C172 E7    48            STB    HCOUNT,U
         C174 86    0D            LDA    #CR
         C176 C6    0A            LDB    #LF        print a CRLF.
         C178 17    0186          LBSR   OUT2CH
         C17B 30    45            LEAX   VAL,U      print the address.
         C17D 17    0221          LBSR   OUT4HS
         C180 86    20            LDA    #' '
         C182 17    0184          LBSR   OUTCH
         C185 AE    45            LDX    VAL,U
         C187 17    0207  HLOOP2  LBSR   OUT2HS     print 16 hex bytes separated by sp.
         C18A 86    20            LDA    #$20
         C18C 17    017A          LBSR   OUTCH
         C18F 6A    48            DEC    HCOUNT,U
         C191 26    F4            BNE    HLOOP2
         C193 CC    2020          LDD    #$2020
         C196 17    016C          LBSR   OUT2CH
         C199 C6    10            LDB    #16        now do the 16 columns of ascii bytes.
         C19B E7    48            STB    HCOUNT,U
         C19D AE    45            LDX    VAL,U
         C19F A6    80    HLOOP3  LDA    ,X+
         C1A1 17    01C0          LBSR   CLASS      print an underline if non-printable char.
         C1A4 27    02            BEQ    OUTACH
         C1A6 86    5F            LDA    #$5F
         C1A8 17    015E  OUTACH  LBSR   OUTCH
         C1AB 6A    48            DEC    HCOUNT,U
         C1AD 26    F0            BNE    HLOOP3
         C1AF AF    45            STX    VAL,U
         C1B1 6A    49            DEC    ACOUNT,U
         C1B3 26    BB            BNE    HLOOP1
         C1B5 30    89 FF00       LEAX   -$100,X    restore address to original.
         C1B9 AF    45            STX    VAL,U
         C1BB 35    04            PULS   B          return the lsb of entered address to orig
         C1BD E7    46            STB    VAL+1,U

                          *PRINT INSTRUCTIONS AT BOTTOM.
         C1BF CC    1600          LDD    #$1600
         C1C2 17    0120          LBSR   LOADCUR
         C1C5 30    8D 02A1       LEAX   HELP,PCR
         C1C9 17    0148          LBSR   PSTRNG

                          ******************************************
                          *  The screen is now formatted for the main program loop   *
                          ******************************************

                          *INPUT A CHARACTER AND PROCESS IT
         C1CC 17    00F6  HEXLOOP LBSR   PORTADR    print the address at top.
         C1CF 17    00FE          LBSR   UPOSIT     position the cursor at the proper screen address.
         C1D2 17    0138  INPUT   LBSR   INCH       get a character w/o echo.
         C1D5 30    8D 0071       LEAX   TBLEND,PCR Search for it in the command table first.
         C1D9 34    10            PSHS   X
         C1DB 30    8D 0052       LEAX   CNTRL,PCR
         C1DF A1    84    HCOM    CMPA   ,X         find a match in table?
         C1E1 27    18            BEQ    FINDH
         C1E3 30    03            LEAX   3,X        point to next entry.
         C1E5 AC    E4            CMPX   ,S         reached end of table yet?
         C1E7 25    F6            BLO    HCOM       no, try next table entry.
         C1E9 32    62            LEAS   2,S        adjust stack.
         C1EB 6D    47            TST    MODE,U     ascii or hex entry?
         C1ED 27    0B            BEQ    HEX11      branch if in hex-entry mode.
         C1EF 17    0172          LBSR   CLASS      check for "printable" ascii character.
         C1F2 26    DE            BNE    INPUT      ignore the char if not.
         C1F4 20    2C            BRA    MEMMOD     modify memory (store the char) if it's valid.

                          *Command is in table, execute it.
         C1F6 32    62    FINDH   LEAS   2,S        adjust stack.
         C1F8 6E    01            JMP    1,X        command table jump.

         C1FA 17    0167  HEX11   LBSR   CLASS      valid hex?
         C1FD 2A    D3            BPL    INPUT      ignore the char if not.
         C1FF 17    0107          LBSR   OUTCH      echo it if it is.
         C202 17    0156          LBSR   ASCHEX     convert ascii hex number to actual hex.
         C205 48                  ASLA              build a hex byte.
         C206 48                  ASLA
         C207 48                  ASLA
         C208 48                  ASLA
         C209 34    02            PSHS   A
         C20B 17    00FF          LBSR   INCH       get the lsb.
         C20E 17    0153          LBSR   CLASS      valid hex?
         C211 2B    07            BMI    HEX12
         C213 35    02            PULS   A          fix stack and screen, because lsb is invalid.
         C215 17    008C          LBSR   UPDATE
         C218 20    B2            BRA    HEXLOOP
         C21A 17    00EC  HEX12   LBSR   OUTCH      echo the lsb.
         C21D 17    013D          LBSR   ASCHEX
         C220 AB    E0            ADDA   ,S+        acca has the new hex char.
         C222 AE    45    MEMMOD  LDX    VAL,U      modify the memory location with the new data.
         C224 A7    84            STA    ,X
         C226 8D    7C            BSR    UPDATE
         C228 E6    46            LDB    VAL+1,U    inc the memory address, only to the end of page.
         C22A 5C                  INCB
         C22B 27    9F            BEQ    HEXLOOP
         C22D E7    46            STB    VAL+1,U
         C22F 20    9B            BRA    HEXLOOP

                          ************************
                          *  COMMAND BRANCH TABLE FOR CONTROL CHARS.  *
                          * This table is searched for the processing *
                          * of the control, and special characters.   *
                          ************************
         C231 08          CNTRL   FCC    08         left cursor.
         C232 20    32            BRA    HLEFT
         C234 0A                  FCC    LF         down cursor.
         C235 20    57            BRA    HDOWN
         C237 0B                  FCC    VT         up cursor.
         C238 20    49            BRA    HUP
         C23A 0C                  FCC    FF         right cursor.
         C23B 20    38            BRA    HRIGHT
         C23D 14                  FCC    20         control T, toggle hex/ascii entry modes.
         C23E 20    20            BRA    TOGGLE
         C240 01                  FCC    1          control A, get new address to dump.
         C241 20    56            BRA    CHSTART
         C243 06                  FCC    6          control F, forward one page.
         C244 20    06            BRA    FPAGE
         C246 02                  FCC    2          control B, backward one page.
         C247 20    0D            BRA    BPAGE
         C249 1B                  FCC    $1B        escape, exit to flex.
         C24A 20    50    TBLEND  BRA    HEXIT

                          * CONTROL-CHAR ROUTINES.
         C24C EC    45    FPAGE   LDD    VAL,U      forward one page.
         C24E C3    0100          ADDD   #$0100
         C251 ED    45            STD    VAL,U
         C253 16    FF00          LBRA   DUMPS

         C256 EC    45    BPAGE   LDD    VAL,U      back one page.
         C258 83    0100          SUBD   #$0100
         C25B ED    45            STD    VAL,U
         C25D 16    FEF6          LBRA   DUMPS

         C260 63    47    TOGGLE  COM    MODE,U     toggle between entry modes.
         C262 16    FF67          LBRA   HEXLOOP

         C265 E6    46    HLEFT   LDB    VAL+1,U    move cursor left.
         C267 5A                  DECB
         C268 C1    FF            CMPB   #$FF
         C26A 1027  FF64          LBEQ   INPUT
         C26E E7    46            STB    VAL+1,U
         C270 8D    32            BSR    UPDATE
         C272 16    FF57          LBRA   HEXLOOP

         C275 E6    46    HRIGHT  LDB    VAL+1,U    move cursor right.
         C277 5C                  INCB
         C278 1027  FF56          LBEQ   INPUT
         C27C E7    46            STB    VAL+1,U
         C27E 8D    24            BSR    UPDATE
         C280 16    FF49          LBRA   HEXLOOP

         C283 E6    46    HUP     LDB    VAL+1,U    move cursor up.
         C285 C0    10            SUBB   #$10
         C287 E7    46            STB    VAL+1,U
         C289 8D    19            BSR    UPDATE
         C28B 16    FF3E          LBRA   HEXLOOP

         C28E E6    46    HDOWN   LDB    VAL+1,U    move cursor down.
         C290 CB    10            ADDB   #$10
         C292 E7    46            STB    VAL+1,U
         C294 8D    0E            BSR    UPDATE
         C296 16    FF33          LBRA   HEXLOOP

         C299 16    FE6D  CHSTART LBRA   MEMEX1

         C29C 32    6A    HEXIT   LEAS   ARGS,S
         C29E 17    005F          LBSR   CLRSCR
         C2A1 7E    CD03          JMP    WARMS

                          *UPDATE BOTH HEX, AND ASCII SCREEN SECTIONS
         C2A4 AE    45    UPDATE  LDX    VAL,U      get address.
         C2A6 A6    47            LDA    MODE,U     save current mode status.
         C2A8 34    02            PSHS   A
         C2AA 6F    47            CLR    MODE,U     set mode to hex first.
         C2AC 8D    72            BSR    UPOSIT     position cursor.
```

```
C2AE 17 00E6         LBSR  OUT2H      output the memory byte.
C2B1 A3 47           CMPD  MODE,U     get mode to ascii.
C2B3 8D 1B           BSR   HPOSIT     position cursor.
C2B5 A6 1F           LDA   -1,X       get char.
C2B7 17 00A6         LBSR  CLASS      see if printable.
C2BA 27 0C           BEQ   HPAEC
C2BC 86 5F           LDA   #$5F
C2BE 8D 40    HPAEC  BSR   OUTCH
C2C0 35 02           PULS  A
C2C2 A7 47           STA   MODE,U
C2C4 39              RTS

              *PRINT HEXADECIMAL ADDRESS ON SCREEN HEADER
C2C5 CC A022  PRNTADR LDD  #$A022
C2C8 17 0027         LBSR  LOADCUR
C2CB 30 45           LEAX  VAL,U
C2CD 16 00D1         LBRA  OUT4H

              *POSITION CURSOR ON SCREEN DEPENDING ON MODE AND "BYTE-ON-PAGE"
C2D0 E6 46    HPOSIT LDB   VAL+1,U
C2D2 57              ASRB
C2D3 57              ASRB
C2D4 57              ASRB
C2D5 57              ASRB
C2D6 C4 0F           ANDB  #$0F
C2D8 CB 05           ADDB  #5
C2DA 34 04           PSHS  B
C2DC E6 46           LDB   VAL+1,U
C2DE C4 0F           ANDB  #$0F
C2E0 6D 47           TST   MODE,U     check if in ascii or hex screen sections.
C2E2 26 07           BNE   ASCMOD
C2E4 86 03           LDA   #3
C2E6 3D              MUL
C2E7 CB 05           ADDB  #5
C2E9 20 02           BRA   DOLOAD
C2EB CB 37    ASCMOD ADDB  #55
C2ED 35 02    DOLOAD PULS  A
C2EF 16 0000         LBRA  LOADCUR

              ************************
              *   LOW LEVEL SUBROUTINES  *
              * these may have to be modified de- *
              * pending on your terminal's controls *
              * character sequences.               *
              ************************

              *(1) POSITION CURSOR AT ACCD (ROW=ACCA),(COLUMN=ACCB).
C2F2 34 06    LOADCUR PSHS D
C2F4 CC 1B3D         LDD   #$1B3D     escape-equal sequence (for televideo).
C2F7 8D 0C           BSR   OUT2CH
C2F9 35 06           PULS  D
C2FB C3 2020         ADDD  #$2020     add offset (for televideo).
C2FE 20 05           BRA   OUT2CH

              *(2) CLEAR THE SCREEN
C300 CC 1B2B  CLRSCR LDD   #$1B2B     escape-plus sequence (for televideo).
C303 20 00           BRA   OUT2CH

              *(3) OUTPUT THE 2 CHARS IN ACCD.
C305 8D 02    OUT2CH BSR   OUTCH
C307 1F 98           TFR   B,A        go into outch routine.

              *(4) OUTPUT A CHARACTER TO TERMINAL
C309 6E 9F D3F9 OUTCH JMP  [OUTCH]    flex vector.

              *(5) INPUT A RAW CHAR W/O ECHO FROM KEYBOARD
C30D 6E 9F D3E5 INCH  JMP  [INCH]     flex vector.

              *(6) PRINT STRING OF CHARS.
C311 17 FFF5  PDAT   LBSR  OUTCH
C314 A6 80    PSTRNG LDA   ,X+
C316 81 04           CMPA  #4
C318 26 F7           BNE   PDAT
C31A 39              RTS

              *(7) CONVERT HEXADECIMAL ASCII STRING TO 16 BIT RIGHT JUSTIFIED # IN D
              * ENTRY:X=STARTING ADDRESS OF STRING
              * EXIT:D=16-BIT VALUE
              ************
C31B 4F       GETHEX CLRA               clear temp stack area.
C31C 5F              CLRB
C31D 34 06           PSHS  D
C31F C6 0D           LDB   #CR        find ending carriage return in string.
C321 A1 85    FINDCR CMPA  B,X
C323 27 03           BEQ   FOUND
C325 5C              INCB
C326 20 F9           BRA   FINDCR
C328 5D       FOUND  TSTB               exit routine with x=0 if no number was typed.
C329 27 2E           BEQ   GETEXIT
C32B 5A              DECB               accb is the string offset pointer.
C32C A6 85           LDA   B,X        do bits 0-3
C32E 8D 2D           BSR   ASCHEX
C330 A7 61           STA   1,S
C332 5A              DECB
C333 2B 24           BMI   GETEXIT
C335 A6 85           LDA   B,X        do bits 4-7.
C337 8D 22           BSR   ASCHEX
C339 48              LSLA
C33A 48              LSLA
C33B 48              LSLA
C33C 48              LSLA
C33D AB 61           ADDA  1,S
C33F A7 61           STA   1,S
C341 5A              DECB
C342 2B 15           BMI   GETEXIT
C344 A6 85           LDA   B,X        do bits 8-11
C346 8D 13           BSR   ASCHEX
C348 A7 E4           STA   0,S
C34A 5A              DECA
C34B 2B 0C           BMI   GETEXIT
C34D A6 85           LDA   B,X        do bits 12-15
C34F 8D 0A           BSR   ASCHEX
C351 48              LSLA
C352 48              LSLA
C353 48              LSLA
C354 48              LSLA
C355 AB E4           ADDA  ,S
C357 A7 E4           STA   ,S
C359 35 86    GETEXIT PULS D,PC

              *(8) CONVERT ASCII HEX TO REAL HEX.
C35B 80 30    ASCHEX SUBA  #$30
C35D 81 09           CMPA  #9
C35F 23 02           BLS   ASC2
C361 80 07           SUBA  #7
C363 39       ASC2   RTS

              *(9) TEST AND CLASSIFY ASCII CHARACTER
              * ENTRY: ACCA=CHAR
              * EXIT: ZERO IF ALPHANUMERIC (Z set) NON-ZERO IF NOT
              *       NEG IF HEXADECIMAL (N set) POS IF NOT
              *****************
C364 34 02    CLASS  PSHS  A
C366 81 7A           CMPA  #$7A
C368 22 1A           BHI   NEITHER
C36A 80 20           SUBA  #$20
C36C 2B 16           BMI   NEITHER
C36E 80 10           SUBA  #$10
C370 81 09           CMPA  #9
C372 23 0A           BLS   HEX
C374 80 11           SUBA  #$11
C376 81 06           CMPA  #6
C378 23 04           BLS   HEX
C37A 1C F7           ANDCC #%11110111
C37C 20 02           BRA   CLASS1
C37E 1A 08    HEX    ORCC  #%00001000
C380 1A 04    CLASS1 SEZ
C382 35 82           PULS  A,PC
C384 1C 03    NEITHER ANDCC #%11110011
C386 35 82           PULS  A,PC

              *(10) OUTPUT 2 AND 4 DIGIT HEX NUMBERS X points to it.
C388 44       OUTHL  LSRA
C389 44              LSRA
C38A 44              LSRA
C38B 44              LSRA
C38C 84 0F    OUTHR  ANDA  #$0F
C38E 8B 30           ADDA  #'0
C390 81 39           CMPA  #'9
C392 23 02           BLS   OUTHI
C394 8B 07           ADDA  #7
C396 16 FF70  OUTHI  LBRA  OUTCH
C399 A6 84    OUT2H  LDA   0,X        output 8 bit number to terminal.
C39B 8D EB           BSR   OUTHL
C39D A6 80           LDA   ,X+
C39F 20 EB           BRA   OUTHR

C3A1 8D F6    OUT4H  BSR   OUT2H      output 16 bit number to terminal.
C3A3 20 F4           BRA   OUT2H

              ************************
              * SYSTEM MESSAGES    (with crlf sequence) *
              ************************

C3A5 41 44 44 52  ADRPRM FCC  'ADDRESS:####',4
C3A9 45 53 53 3A
C3AD 24 24 24 24
C3B1 04
C3B2 20 20 20 20  DUMPM  FCC  '      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F        ASCII'
C3B6 20 20 30 20
C3BA 20 31 20 20
C3BE 32 20 20 33
C3C2 20 20 34 20
C3C6 20 35 20 20
C3CA 36 20 20 37
C3CE 20 20 38 20
C3D2 20 39 20 20
C3D6 41 20 20 42
C3DA 20 20 43 20
C3DE 20 44 20 20
C3E2 45 20 20 46
C3E6 20 20 20 20
C3EA 20 20 20 20
C3EE 41 53 43 49
C3F2 49
C3F3 0D              FCB   CR
C3F4 0A              FCB   LF
C3F5 20 20 20 20     FCC   '    / / / / / / / / / / / / / / / /',4
C3F9 20 20 2F 20
C3FD 20 2F 20 20
C401 2F 20 20 2F
C405 20 20 2F 20
C409 20 2F 20 20
C40D 2F 20 20 2F
C411 20 20 2F 20
C415 20 2F 20 20
C419 2F 20 20 2F
C41D 20 20 2F 20
C421 20 2F 20 20
C425 2F 20 20 2F
C429 04

C42A 5E 46 3D 66  HDID   FCC  '^F=forward       ^B=backward        ESC exit'
C42E 6F 72 77 61
C432 72 64 20 20
C436 20 20 20 20
C43A 20 20 5E 42
C43E 3D 62 61 63
C442 6B 77 61 72
C446 64 20 20 20
C44A 20 20 20 20
C44E 20 20 20 45
C452 53 43 20 65
C456 78 69 74
C459 0D              FCB   CR
C45A 0A              FCB   LF
C45B 5E 41 3D 6E     FCC   '^A=new address   ^T=toggle hex/asc',4
C45F 65 77 20 61
```

```
        C463 64 64 73 65
        C467 73 73 20 20
        C46B 20 20 3A 34
        C46F 30 10 6F 67
        C473 67 6C 65 20
        C477 60 65 70 2F
        C47B 61 73 63 64
445
446     C47F 00           MACSP  FCB    05
447     C480 2A                  FCC    '*'
448     C481 00                  FCB    05
449     C482 04                  FCB    4
450
451                              END    MEM13

0 ERROR(S) DETECTED

SYMBOL TABLE:

[illegible] C303  [illegible] 0000  ASC7   C3A3  ASCNS1 C350  ASCNUM C2E1
BACKSP C47F  [illegible] C13C  [illegible] C356  BS  [illegible]  CHS1UP C299
CLASS  C364  CLASS1 C380  CLROCR C380  CONTRL C233  CR     000D
[illegible] C2E0  [illegible] C156  [illegible] C382  FF     000C  FINDCR C321
FINDM  C1F6  FOUND  C328  FPAGE  C24C  GETEXI C359  GETHEI C318
GOODCH C1A8  HCOM   C1DF  HDIR   C42A  HDOWN  C28E  HEI    C37E
HEI12  C21A  HEI11  C29C  HEXLOD C1CC  HEXXX  C1FA  HLEFT  C265
HLOOP1 C170  HLOOP2 C187  HLOOP3 C19F  HPOSIT C200  HRIGHT C275
HUP    C283  INADR  C120  INCH   C300  INCDUN 0000  INPUT  C1D2
LF     000A  LINEBU 0000  LOADCU C2F2  MEMEI  C100  MEMEI1 C107
MEMMOD C222  MODE   0007  NEITHE C384  OCOUNT 0009  OUT2CH C305
OUT2H  C399  OUT4H  C3A1  OUTCH  C309  OUTHI  C396  OUTHL  C388
OUTHR  C38C  PDAT   C311  PRNTAD C2C5  PSTRNG C314  RETU   C14D
START  C105  TBLEND C24A  TOGGL1 C260  UPASC  C28C  UPDATE C266
```

NEWSRELEASE

FOR RELEASE: IMMEDIATELY
CONTACT: JIM LOE (602) 991-1657

SYSTEMS DESIGNWARE announces AutoComm, a new modem program for 6809 FLEX systems. The program automates many of the repetitive functions encountered while working with time sharing.*
Features: A Ao-Dialing, Auto-Redialing, Auto Sign on.

Capabilities include:

1) Send text file from personal computer disk to remote mainframe computer.
2) Save incoming text to disk file (verifys acceptance of XON/XOFF controls).
3) Slow transmission mode based on character verify for systems which require speed below baud rate.
4) Echo, no-echo, add-line-feed, delete-line-feed modes.
5) Keyboard control of printer ON/OFF.
6) Eight software selectable UART modes; 8 bit, 7 bit.
7) Single mode design, with commands prefixed by a comma.
8) Smart command interperter accepts English-like commands and all abbreviations of commands.
9) Self adapts to amount of memory in your computer.
10) Runs in as little as 12K bytes or up to 65K bytes.
11) Reads and writes files of phone numbers to be dialed.
12) Makes any modem a smart modem.

A total of 25 commands with tree structure 'HELP' system.

Useful for conventional time-sharing or electronic mail:
seventeen page manual provided with partial listing showing all I/O (indirect) addressed, time constants, processor clock speed compensation, answer delay: everything necessary to adapt to almost any 6809 FLEX* based system.

Detailed instructions supplied for low cost autodialer interface which allows any modem to do smart modem functions.

Now $79. Visa or Mastercharge, order toll free 800 272-4817.
Manual only: $3.95.

*FLEX is a trademark of TSC Co.

SYSTEMS designware

6712 EAST PRESIDO ST. • SCOTTSDALE, AZ. 85254

Speech Systems

38 W 255 Deerpath Road Batavia, Illinois Tel. (312)879-6880

October 14, 1982

For Immediate Release

Four Voice Music Compiler for Color Computer

SPEECH SYSTEMS, a manufacturer of high quality speech and music synthesizers is proud to annouce The COMPOSER, a 4 voice music compiler for the TRS-80C Color Computer.

No other music compiler available for the Color Computer delivers such outstanding features. The COMPOSER allows one to write music using 4 separate voices over a 7 octave range. The user may specify the tempo and key and may even have these change as the music is played. The program is menu driven making the COMPOSER easy to use. The COMPOSER supports dotted and double dotted notes as well as three types of triplett notes.

As the music plays, a colored kaleidoscope pattern and the number of the note being played is displayed. This feature is particularly important in debugging music.

Compiled music is saved as a completely independent machine language subroutine that may be called and executed by other programs. This allows integrating music and sound effects into other BASIC programs.

A complete manual is included which describes the use and operation of the COMPOSER. Enclosed are several examples to aid the user in getting started.

The COMPOSER requires Extended BASIC and is available for cassette ($24.95) as well as disk users ($29.95).

EMC

11 November 1982

68 Micro Journal
Box 849
Hixson, TN 37343

Dear Don:

The November 1982 Micro Journal contains a letter from Anthony Gasbarre describing modifications which allow the Creative Micro Systems 9617 EPROM Programmer to be used with FLEX. This letter contains an error with respect to use of 32K EPROMs.

The 9617 was designed for use with TMS2516 and TMS2532 EPROMs. Unfortunately, 2716 and 2732 EPROMs have become far more common.

A 2516 and a 2716 are effectively identical. Therefore, there is no problem in using the 9617 with 2716 EPROMs, and using the supplied 2516 software. However, the 2732 and 2532 are somewhat different chips. They are pin compatible with the following exceptions:

1. A11 The highest order address line must be transposed from pin 18 of the 2532 to pin 21 of the 2732.

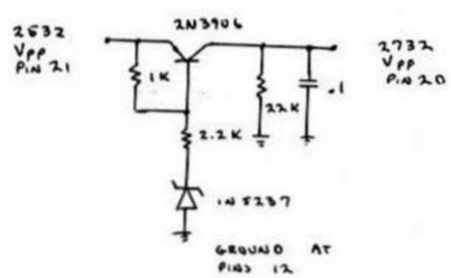
2. CS This is equivalent to CE on the 2732, and must be moved from pin 20 of the 2532 to pin 18 of the 2732.

3. Vpp This is the programming voltage, which must be +25 during the programming of both types of EPROMs, but +5 during read operations on the 2532, and 0 volts during reading of the 2732. This required design of a level shifter circuit, drawn below.

An adapter was constructed on a small piece of perf board, with a Textool 24 pin zero insertion force socket on top, and a 24 pin DIP header mounted below. The header is inserted into the socket on the 9617, and the 2732 inserted into the new ZIF socket. Except as indicated above, corresponding terminals on the socket and the header are wired together. With this adapter, the CMS 2532 software and hardware can be used with the more common 2732 EPROMs.

Sincerely,
ENERGY MANAGEMENT CORPORATION

Michael Hirsch
Manager, Electronic Systems Division



IMPROVED PERFORMANCE
CMOS 14411 BAUD RATE GENERATOR

(December 1, 1982) (Sunnyvale, California)--Zytrex introduces a pin-for-pin, function-for-function, direct replacement for the 14411 baud rate generator. The new ZT14411 contains a crystal oscillator and frequency division circuitry to simultaneously generate 50% duty cycle

square waves for the following baud rates: 75, 110, 134.5, 150, 200, 300, 600, 1200, 2400, 3600, 4800, 7200, and 9600 baud. Multiples of 1, 8, 16, and 64 times the baud rates are selectable by two input pins. Besides these baud rates, also concurrently output is the crystal frequency and one half the crystal frequency; these outputs could be used to provide a clock for microprocessors or other devices.

The ZT1441l bit rate generator is constructed in CMOS with fully LS-TTL compatible inputs and outputs. The output levels go from ground to VCC and have increased drive for a minimum fanout of 4 low-power Shottky TTL loads. This means that for many applications the conventional buffering through inverters could be eliminated. The inputs which select the frequency multiple, also accept either TTL or CMOS levels.

The ZT1441l is fully static so it could be used as a frequency divider at frequencies down to DC, and when using the normal 1.8432 MHz crystal the maximum power supply current, with no output drive loading, is less than 15mA at 5 volts.

The ZT1441l is available in 24-pin plastic or ceramic dual-in-line configuration in commercial or industrial temperature ranges.

For quantity pricing contact La Parr Stuart at (408) 733-3973.

NEW PRODUCT ANNOUNCEMENT - FOR IMMEDIATE RELEASE

Expansion Interface for the TRS-80 Color Computer

Color Computer users can now have power rivaling the Apple-II. The new General Automation CX-2001A Expansion Interface provides a Centronics compatible parallel I/O port, a 64K Memory Access circuit, and I/O expansion capability for up to 7 additional peripheral cards. An impressive array of specialized I/O cards will soon be available - Votrax Speech Synthesizer, Quad Parallel I/O, 12-bit A/D, TV Digitizer, and more!

The General Automation interface requires NO modification whatever to the Color Computer. The existing 24K Basic ROM, as well as the cassette, serial, and joystick ports remain available. The Expansion Interface is compatible with the Radio Shack Disk System. The 64K Memory Access circuit allows 32K Rev-E Color Computers to double their available RAM.

In addition, an aluminum chassis is available. The chassis, also ideal for stand alone use, provides support for a television and is the system enclosure for the interface electronics.

The CX-2001A Expansion Interface and the CX-3001A Chassis may be purchased separately. A special introductory package, the CX-P1, includes both and retails for $189.95.

The Color Computer of the Future...

With the General Automation Expansion Interface

After you've played a few games with your Color Computer, what next? It is becoming increasingly apparent that the Color Computer is not a toy. But it does have one serious drawback. If you used a disk controller or ROM cartridge, it was almost impossible to add anything else - until now. General Automation is taking the lead in developing a line of sophisticated add-on products for the Color Computer. The new CX-2001A Expander Card and the CX-3001A Aluminum Chassis are impressive additions to your computer.

The Expander Card attaches to any Color Computer with NO MODIFICATION. As soon as you plug it in, you immediately have a Parallel Printer Port, 64K Memory Access, and a buffered Expansion Bus. You still have FULL USE OF THE RADIO SHACK DISK SYSTEM, the cassette, and the joysticks - everything. What's more, the serial port remains available for use with a modem or even another printer. Imagine - your computer can talk to a MODEM AND A PRINTER AT THE SAME TIME!

With all this new power, you may have some questions. Let's see if we can answer some of them...

What is the 64K Memory Access Circuit?

The Expander Card has a 64K memory access circuit which allows 32K (Rev-B) computers to double their available RAM. Though not widely publicized, 32K Rev-E machines actually contain 64K RAM chips. Normally, the Color Computer can access only 32K of this RAM. With our circuit, it is possible to access almost the entire 64K.

To the BASIC programmer, there is now an additional 8K (16K if no disk) of RAM for use with PEEK, POKE and machine language subroutines. Machine language programs can use it all. This circuit provides enough RAM for memory based operating systems such as FLEX. Now many powerful languages and software packages become available to Color Computer users!

"CHEAP TALKER" Correction

We received a phone call from John R. Kelty noting a few changes on his article "CHEAP TALKER" published in the December 1982 issue on page 18. Not shown on the diagram on page 19 please note;
There should have been a circle 6 out of the computer to pin 25 (E Clock) on the I.C.
Also;
RS0 which is pin 36 on I.C. should go to pin 19 (A0) on the computer.
RS1 which is pin 25 on I.C. should go to pin 20 (A1) on the computer.
The final change given to us was that line 1030 on page 19 should have actually read;
1030 IF (PEEK(A+1)AND128)THEN1030 ELSE RETURN 'RETURN WHEN A/R GOES HIGH
The difference between the above line 1030 and the line 1030 as published is that in the middle of the line the words RETURN ELSE 1030 have been reversed to read 1030 ELSE RETURN.

CLASSIFIED ADVERTISING

Odds & Ends. Call for info. Expandor printer, drive cabinet, keyboard, memory, UDB-1, PMB-1, MPA, CPU-2.
MAF 512-687-7030

SWTPC 6809 S Box with 128K, 4 Serial Ports, 1 Parallel Port. 4 8212 SWTPC Terminals. DMF2 w/controller. CDS-1 Harddisk. Uniflex Operating System w Basic, Precompiler, Text Processor etc. Make me an offer.
LEW days(615)842-4607 eve.(615)842-3353

Wanted: Working MP-A2 card or bareboard. MP-8M, MP-40 and clock calender card.
P. Smith, 2958 Orillia, Victoria, B.C., Canada V9A 1Y5 (604)382-6246

Grab bag special! 2 JPC Products CK-7 clocks, SWTPC AC-30 cassette interface, cables & documentation, & MP-AL 6800 processor board (all working); plus 2 MP-4M memory boards with problems. $125.00 buys all. Will trade for working MP-T or MP-L2 or MP-S2 or 2 MP-S interfaces. Don't make me give them away.
Dwight Lanpher, Box 472 Summit Rd, N.E. Harbor, ME 04662

For Sale: Small computer manufacturing business for Motorola 6800/6809 Exorciser Buss, includes drawings, artwork, printed circuit boards, chassis and many parts. Designs include 6800 and 6809 CPU boards, acia, memory, wire wrap, and mother boards. Disk controller board will run 5 1/4 TSC FLEX. Also Motorola Exorciser with 6800/6809 with 64K ram. Exorterm with 6809 CPU and disk controller card; Exordisk systems, ADM 3 terminal. Will sell all or any part.
Contact: John Ranson, P.O. Box 11451, Winston Salem, N.C. 27106 Phone: (919)722-1596

6800/6809 CPU card, AC-30, Percom cls 30+, 32K mem, 6809 Software.
Ben Bezenek (612)733-3650

FOR SALE: SSB 6809 Micro with OS9. 32K, DCB 4A, Dual 8 in. disks, 4 serial ports, 2 parallel ports. Software included OS9 level 1, Basiconline. ASMX, editor, super debugger. $4200 retail value-1 yr old. $3000 or best offer.
Harry Fair, 1952 S. Holland, Lakewood, CO 80227. (303)773-1313, 985-7419

SWTPC 6800 w/44K, MPC, MPS, MPL, Clock and Disk Controller. AC30, ACT-1 Terminal, MT80P Printer Disk Software. $900 or best. Jim Hall, 2063 108th Ave., Otsego, MI 49078. (616)694-9760

Gimix Mainframe, 6809 CPU "Plus", 5/8 controller, 56K, 2-892 MPI drives (DS,DD,96TPI), serial and parallel cards, 80 x 24 video board, keyboard w/enclosure, Xbasic, precompiler, sortmerge, modem, editor(s), OS-9/Flex prom switch. $3,500 or best.
Call 312-266-6809 ask for Les Multack

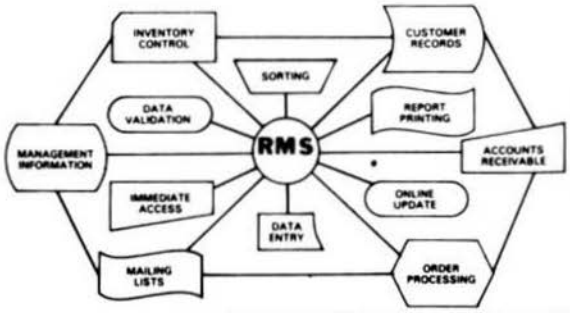
Gimix mainframe, 6800, 32K, 2-FD-8 8" disk drives w/controller, serial and parallel cards, prom board, lots of parts, G/L, Basic, $2,800 or best offer. Also full accounting package available.
CaLL 312-266-6809 ask for Les Multack

SWTPC 6800 Computer: Chassis, Motherboard and Power Supply, $100; MPA2, $100; MF-68, $600 as is; MPS, $30; CT-64, $100 as is; ACT-1A Terminal, $175.
John Current (301)977-0507 7-9PM

6809 RECORD MANAGEMENT SYSTEM RMS

DATABASE MANAGEMENT

- USER DEFINED RECORD FORMAT VIA DATA DICTIONARY
- SCREEN ORIENTED, FORM FILL OUT TYPE OF ACCESS
- OPTIONAL TWO LEVEL RECORD HIERARCHY
- ALL FILES IN ASCII TEXT FORMAT, BASIC COMPATIBLE
- DIRECT ACCESS BY KEY FIELD, MULTIPLE INDEX FILES
- EXTENSIVE DOCUMENTATION, SAMPLE APPLICATION
- VERSATILE, PROFESSIONAL QUALITY REPORT WRITER
- BUILT-IN SORT/MERGE
- EASY TO USE



RMS is a complete DATABASE MANAGEMENT package for the 6809 computer. It is made up of five machine language programs that make up the most powerful business programming tool available for the 6809. It can be used by the relative novice, to implement an incredible variety of information storage and retrieval applications, without any programming. However, the programmer can use RMS as part of the solution to a larger problem, saving many hours of unnecessary program development time. RMS can be used to handle data input, editing, validation, on-line retrieval, sorting and printed reports. Custom data manipulation can be filled in by the user's BASIC programs.

SINGLE CPU LICENSE

FLEX*	$200
OS-9+	$250
UNIFLEX*	$300

TERMS: VISA/MC/PREPAID

WASHINGTON COMPUTER SERVICES
3028 SILVERN LANE
BELLINGHAM, WA 98226
1 (206) 734-8248

In Australia & Southeast Asia, Available Through:
Paris Radio Electronics, 7A Burton St.
Darlinghurst, NSW 2010 Sydney Australia

* FLEX and UNIFLEX are trademarks of Technical System Consultants Inc.; + OS-9 is a trademark of Microware

Master Control For Your Computer
JOB CONTROL PROGRAM

by Peter Murray

Features and Applications

- parameter substitution
- conditional branching/loop control
- error trapping and recovery options
- built-in text editor
- compact . . . all commands reside in 2K bytes
- co-resident with executing programs
- fast, efficient machine language implementation
- both 6809 and 6800 versions available
- runs on all standard FLEX computers, including TRS-80 Color Computer*
- compatible with all standard FLEX programs
- fully supported by the author
- comprehensive, well written 80-page manual with relevant examples
- source code available for customization
- liberal license arrangement for software producers
- make complex processing routines simple
- perform file maintenance, backups easily
- software producers: make systems user-friendly, easier to use and operate
- computer dealers: demonstrate software/hardware automatically
- simplify program development activities — allow your computer to run unattended for long compiles, assemblies, etc.

ORDERING INFORMATION

- JCP is available on 5 1/4" and 8" diskette for 6800 and 6809 FLEX computers (specify both)
- Object code only, $29.95 (special price good for orders received by Feb. 28, 1983)
- Object + source, $89.95
- Manual only, $12.95 (credited toward purchase)
- Please add $3.00 SH charges
- Colorado residents add X% state sales tax

ChiraTech Scientific Instruments
204 N. Link Lane, Alpha 9
Fort Collins, Colorado 80524
(303) 484-1913

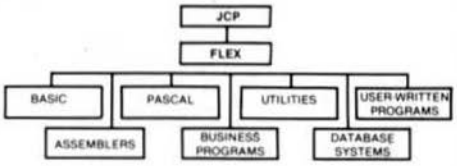
A New Year Special Price For JCP:

$29.95

(Object code on FLEX diskette, plus 60-page manual)

See July 1980 '68' Micro Journal review of JCP

JCP Coordinates your FLEX computer



What is JCP?

JCP, field-tested by satisfied users for over two years, is a program which loads into memory, then controls operation of the computer. Sequences of FLEX programs, utilities, language processors, etc. are executed, with JCP supplying all parameters, options, and operator inputs as required (or, allowing direct operator input, if desired). You define a JCP procedure (job stream) once; thereafter, you type a simple one-line command to initiate the job. You don't have to remember all those operational details required to run a routine job. Just tell JCP to run a procedure. JCP even handles error situations under user options — JCP can handle the error or can BREAK to give you the chance to look at the situation, take corrective action, then CONTINUE the procedure from the point of interruption! JCP allows conditional branching within a job stream. JCP will substitute parameters into the job stream, allowing general purpose procedures to handle complex compiles, assemblies, link-edits, sorts and so forth. JCP puts you in control of your computer!

Coming Soon . . .

- Buffered serial interface (SS-30)
- Buffered Centronics Parallel interface (SS-30)
- 64K (expandable to 128K) Print SPOOler (SS-30)
- . . . plus other hardware and software products for 68XX computers!

Trademark Credits

FLEX is a registered trademark of Technical Systems Consultants, Inc. TRS-80 Color Computer is a registered trademark of the Tandy Corp. *Frank Hogg Laboratory supplies a version of FLEX which runs on the TRS-80 Color Computer.

TEN MOST-ASKED QUESTIONS

ABOUT DYNACALC™

THE ELECTRONIC SPREAD-SHEET FOR 6809 COMPUTERS

1. **What is an electronic spread-sheet, anyway?**
   Business people use spread-sheets to organize columns and rows of figures. DYNACALC simulates the operation of a spread-sheet without the mess of paper and pencil. Of course, corrections and changes are a snap. Changing any entered value causes the whole spread-sheet to be re-calculated based on the new constants. This means that you can play, 'what if?' to your heart's content.

2. **Is DYNACALC just for accountants, then?**
   Not at all. DYNACALC can be used for just about any type of job. Not only numbers, but alphanumeric messages can be handled. Engineers and other technical users will love DYNACALC's sixteen-digit math and built-in scientific functions. There's even a built-in sort command, so you could use DYNACALC to manage small data bases - up to 256 records.

3. **What will DYNACALC do for ME?**
   That's a good question. Basically the answer is that DYNACALC will let your computer do just about anything you can imagine. Ask your friends who have VisiCalc, or a similar program, just how useful an electronic spread-sheet program can be for all types of household, business, engineering, and scientific applications.

4. **Do I have to learn computer programming?**
   NO! DYNACALC is designed to be used by non-programmers, but even a Ph.D. in Computer Science can understand it. Built-in HELP messages are provided for quick reference to operating instructions.

5. **Do I have to modify my system to use DYNACALC?**
   Nope. DYNACALC uses any standard 6809 configuration, so you don't have to spend money on another CPU board or waste time learning another operating system.

6. **Will DYNACALC read my existing data files?**
   You bet! DYNACALC has a beautifully simple method of reading and writing data files, so you can communicate both ways with other programs on your system, such as the Text Editor, Text Processor, Sort/Merge, RMS data base system, or other programs written in BASIC, C, PASCAL, FORTRAN, and so on.

7. **How fast is DYNACALC?**
   Very. Except for a few seldom-used commands, DYNACALC is memory-resident, so there is little disk I/O to slow things down. The whole data array (worksheet) is in memory, so access to any point is instantaneous. DYNACALC is 100% 6809 machine code for blistering speed.

8. **Is there a version of DYNACALC for MY system?**
   Probably. You need a 6809 computer (32k minimum) with FLEX or UniFLEX operating system. A version for OS-9 is also in the works. You also need a decent CRT terminal, one with at least 80 characters per line, and direct cursor addressing. If your terminal isn't smart enough for DYNACALC, you probably need a new one anyway. The UniFLEX version of DYNACALC also allows you to mix different brands of terminal on the same system. There's also a special version of DYNACALC for Color Computers equipped with FLEX and Data-Comp's F-MATE. A version for Frank Hogg's Color Computer FLEX is also being done.

9. **How much does DYNACALC cost?**
   The FLEX versions are just $200 per copy; UniFLEX version $395. Foreign orders add $10 per copy for postage. We encourage dealers to handle DYNACALC, since it's a product that sells instantly upon demonstration. Call or write on your company letterhead for more information.

10. **Where do I order DYNACALC?**
    See your local DYNACALC dealer, or order directly from CSC at the address below. We accept telephone orders from 10 a.m. to 6 p.m., Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Please specify diskette size for FLEX versions. Software serial number is required for the UniFLEX version of DYNACALC.

ORDER YOUR DYNACALC™ TODAY

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 7A Burton St., Darlinghurst, NSW 2010 Sydney. Telephone: 02-357-5111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

CSC

Computer Systems Center

13461 Olive Blvd.

Chesterfield, MO 63017

(314) 578-5020

UniFLEX software prices include maintenance for the first year.

DYNACALC, DYNAMITE, and DYNAMITE + are trademarks of Computer Systems Center.

F-MATE is a trademark of Data-Comp.
VisiCalc is a trademark of VisiCorp.
OS-9 is a trademark of Microware and Motorola.
FLEX and UniFLEX are trademarks of TSC.

ALSO FROM CSC

DYNAMITE +
"THE CODE BUSTER"

now available for UniFLEX
OS-9 version soon

DYNAMITE + is a new version of DYNAMITE, our popular 6809/6800 disassembler package for 6809 FLEX. Present users of DYNAMITE can upgrade to DYNAMITE + by sending us the original DYNAMITE diskette and $40 (plus $5 for foreign postage). DYNAMITE + does everything DYNAMITE does, and more! A cross-reference generator has been added, label files are now maintained only in text form (LABEL EQU $xxxx), and boundary file specifications have been tremendously simplified, which makes it easier to disassemble large programs containing lots of big tables.

The UniFLEX version of DYNAMITE + does everything the FLEX version does, and also automatically handles system calls and 'info' areas.

DYNAMITE + is available for $100 per copy on FLEX (specify diskette size), and $300 on UniFLEX. Foreign orders add $5 per copy for postage.

SCREEN EDITOR/WORD PROCESSOR FOR DOS6x, FLEX, OS-9

IF YOU ARE CONSIDERING UPGRADING YOUR EDITING AND PROCESSING CAPABILITIES, HERE ARE A FEW GOOD REASONS TO CONSIDER SCREDITOR...

REASON #1 - SCREEN EDITING

Screen editing (sometimes called cursor-based editing) is the most efficient way in which to enter your thoughts into the computer. Since you see the actual text you are entering on the screen as if you were looking at a printed page, organization becomes a matter of observation rather than guesswork.

REASON #2 - DYNAMIC SCREEN FORMATTING

With dynamic screen formatting, the screen becomes an image of what the completed job will look like. Within the capabilities of the display terminal itself, you are constantly looking at what you would have if your job were printed. Every change you make, every addition or deletion, is automatically incorporated in a fully-justified, ready-to-print form. You no longer need to guess what all those cryptic processing commands you may have used with other software will turn your finished job into!

REASON #3 - NO MACHINE LANGUAGE PATCHING

SCREDITOR III is so easy to use that you NEVER need to make machine language patches to integrate it into your system. It [illegible] a series of questions about your printer and terminal's capabilities. Installation is complete.

REASON #4 - NO EXTRA "HIDDEN" PROGRAM COSTS

SCREDITOR III is complete as purchased. You get screen editing, formatting, mailing list merging and much more, all in a single program. Add it up. Even though the competition may provide MOST of the features of SCREDITOR III, how much will you pay? And, more important, how many programs will you have to buy?

REASON #5 - OPERATING SYSTEM SUPPORT

To our knowledge, SCREDITOR III supports more processor and operating system combinations than any other single editing and word processing package available. OS-9 (level one and two), Flex and SSB DOS, 6800 and 6809 versions are available. Except for those commands which are peculiar to a particular operating system, SCREDITOR III is identical in operation in all versions.

REASON #5 1/2 - OPERATING SYSTEM SUPPORT COSTS

If you purchase another company's product for your FLEX-9 system and decide to run OS-9, how much will it cost you for the extra copy of their software (if it's available)? See reason #5. If with SCREDITOR III, tell us you are upgrading and pay [illegible] of the price you paid for the first copy and you can have the same great capabilities with both operating systems!

REASON #6 - CAPABILITIES

Not only is SCREDITOR III the only COMPLETE editing and word processing program available, it's also the most powerful. See if the other combinations of programs available give you true multi-column editing. Or allow you to add and decimal align rows and columns of numeric data. Or allow you to change margins and tabs at any time during editing just like a typewriter. Or give you commands to re-justify and change the size of paragraphs and lines on-screen, while you watch. Or edit and re-execute commands. Or allow essentially unlimited keyed text and command entry replacement with a single keystroke (macros). Or let you save all of those macros to a data file to be re-loaded for use in later edit sessions. Or edit your macros. Or...

REASON #7 - DOCUMENTATION

SCREDITOR III has the most comprehensive documentation available for any 68xx editor/word processor. The Technical Reference Manual is currently about 100 pages long. The Training and User's Guide we are now preparing will be about another 100 pages. We're also working on our own Tutorial Cassette [illegible]. We even provide a HELP command which gives you the syntax of all of SCREDITOR III's commands and at least one paragraph of description on each!

REASON #8 - PRICE

Considering its capabilities, SCREDITOR III is the least-expensive program of its type available today. Period.

REASON #9 - HAPPINESS

We want your business. Think how happy you will make us if you decide to buy SCREDITOR III!

The following information should answer most of your questions about SCREDITOR III. If you have additional questions, please call during our office hours, or, better still, order the manual (when you decide to buy SCREDITOR III, we'll apply your manual purchase against the software price).

PROCESSORS SUPPORTED	- 6800, 6809
OPERATING SYSTEMS	- FLEX, SSB DOS, OS-9 (level I and II)
DISPLAY REQUIREMENTS	- Serial terminal with computer-addressable cursor. 4800 baud minimum data rate, 9600 or better preferred. Clear EOL and clear EOS not required but highly desirable. Insert and Delete line improve display speed. Suggested terminals include Televideo, Heath/Zenith H-19, SWTPCo. Virtually all addressable-cursor terminals can be used. ANSI terminals supported by the time you read this. Memory-mapped displays require software to provide terminal look-alike capabilities. Sample software included with FLEX and DOS6x orders. Minimum display size is 16x64. Maximum is 255x255. No software patching is required to adapt software to display size. Unique configurations may be supported by calls to external user-supplied sub-routines.
KEYBOARD REQUIREMENTS	- ASCII character set with control codes required. Function keys generating multiple-character sequences also supported. Control code and function key assignments user-defined and changeable at any time without patching.
KEYBOARD INTERFACE	- PIA parallel, ACIA serial, or external keyboard handlers. Interrupt-driven operation (queue-buffered type-ahead) is a standard user-selected option.
PRINTER HARDWARE	- Current version supports standard printers. Specialty printer capabilities controllable by transmission of control or escape sequences imbedded in text. Underline, boldface, double width simulated on printers with backspace capability. Proportional printing available soon.
PRINTER INTERFACE	- Program uses existing user-supplied software to transfer characters from SCREDITOR III to printer hardware by calls to user sub-routines.
COMMAND SET	- Fourty-six commands (or more by the time you read this) with multiple options on many commands. Thirty-one single key commands for often-used operations such as cursor movements, inserts, deletes, paragraph formatting, etc...
OPERATION AND DISPLAY	- Text entry by 'window-type' screen/cursor editing. Commands by single-key or line entry. Continuous display of current tabs/margins, edit modes, cursor row and column, current line number, horizontal display range.
REQUIRED MEMORY	- (program memory) - Program size - approximately 20 kilobytes. Minimum memory for useful editing - 32 kilobytes.
DISK MEMORY	- Configuration files (3 minimum) - 1 to 2 sectors each HELP file - disk-resident, approximately 19 kilobytes

**************************** AT LAST! ****************************

AVAILABILITY	- Please allow one to two weeks for delivery. Disks are built as orders are received to allow the most current possible revision of software to be shipped.
PRICE	- SCREDITOR III manual, disk . . . . $150.00 manual only (price may be applied to software order) . . . . . $ 40.00

IMPORTANT INFO

We accept Mastercharge and VISA credit card orders, personal and business checks drawn on U.S. banks and in U.S. dollars, and COD's to the continental United States. Domestic orders shipped by United Parcel Service unless otherwise requested. Overseas orders shipped by air parcel. COD's shipped by United Parcel Service only. Open accounts to government agencies, educational institutions, established accounts and Fortune 1000's only. Virginia residents add 4% state sales tax. On orders under $100, add $5 for shipping and handling. On overseas orders add $10 for shipping.

THE BEST NEWS!

In our haste to do business, too often the temptation to compromise our faith weighs heavily. Even though we have forgiveness when we confess our sins from a God who loves us, we would like to take a moment to ask forgiveness from any of our customers whom we have slighted. Orders do get lost or mis-shipped, and mistakes do happen. If we fail to completely please any of you in either our response to your order, inquiries, or for whatever reason, let us know. We will do anything in our power to satisfy your problems. We thank all of our customers for your prayers, and we pray for each of you that you will come to know the love of God in your lives that we have found!

CoCo SPEECH SYNTHESIZER

The Color Computer is one of the best things to happen to computing since the SS-50 bus! At the price, there is no better way to get into personal computing, and we are now offering a speech synthesizer for the CoCo! With most of the same features of our VS-1, the SP-1 can give your CoCo the power of unlimited speech. Unlike software approaches to speech, the SP-1 does not eat your memory, and provides the clearest, easiest to understand speech we have heard!

The SP-1 is a completely assembled and tested unit which plugs directly into the ROM-PACK slot on the side of the computer. Unlike the VS-1, however, no external speaker is required, as the SP-1 routes its audio directly through the computer and into your television monitor!

We also have a version of our VOX-EDITOR to let you build your voice messages quickly and easily, and then save them to cassette when you are done. The speech files can later be read by Basic in your application programs! The comprehensive manual we supply gives you all the information you need to make your CoCo talk!

SP-1 pack and manual $179.95
SP-1 manual only (price applied to SP-1 purchase) . . . . . $ 19.95

TRANSLATOR FOR SP1

Our SC-1/COLOR translator uses the same highly accurate algorithm as our SC-1 for the VS-1, but drives the SP-1 on the color computer. The SC-1/COLOR, however, is even easier to use! When the SC-1/COLOR translator is loaded, it becomes a part of Basic! To translate and speak English text, numeric data (even punctuations if you desire), you simply program PRINT #1 followed by your text. The translator takes it, converts it to a phoneme stream, and drives the SP-1 directly!

SC-1/COLOR cassette and manual (requires SP-1 for operation) $29.95
(manual not available separately)

SSB UTILITIES DISK

Two of the shortcomings of SSB DOS are that, one, with six-character file names and default extensions it's hard to know from a name what is in a file, and, two, with no disk information sector the only way to know what is on a disk is to list the files! Since we couldn't conveniently make the file names longer, we did the next best thing and wrote a family of programs to allow you to "name" your disks!

Utilities #1 provides a NAME utility to build a disk information sector on each newly-formatted disk. UPDATE lets you change this sector. LOCK lets you write, delete or even LIST protect the files on the disk. LIST is a better listing utility which reads the disk access code from the information sector and will only list LOCK'ed files if you supply the access code. A number of other programs are also included on the disk.

The manual provides information about the way SSB disks are formatted and how the chained sector method work in addition to complete descriptions about each of the programs provided on the disk!

UTILITIES #1 disk and manual $29.95
(manual not available separately)

FANTASTIC PROOFREADER

We are proud to offer Pete Stark's Spell 'n Fix proofreader. We examined a number of very good pieces of spelling correction software. We compared ease of use, speed, correction capability, dictionary maintenance and expansion, disk space overhead, etc..., and found that Pete Stark of StarKits has done a remarkable job on this program. Look at these features and make your own decision....

MULTIPLE DICTIONARIES	Spell 'n Fix allows you to create multiple dictionaries. In this way, if you want, you can put technical and special words in a separate dictionary. Then, when you proofread your documents, select the dictionaries which have the words you need, thus speeding up the job!
EXECUTE DOS COMMANDS	Need to list your disk to decide which dictionary or text file to proofread? Want to delete an old file? View a file? Most of your DOS commands can be executed directly from within Spell 'n Fix.
MULTIPLE CORRECTION OPTIONS	With Spell 'n Fix, you can add new words to your dictionary at any time. Or, if you want, you can create a new dictionary for words not found in the existing dictionaries. Or simply print out the new words. Or even ignore them! CORRECTION OR MARKING - With Spell 'n Fix, you can mark incorrectly-spelled words in your text for later correction, or you can tell Spell 'n Fix to correct them as they are found. The choice is yours!
COMPRESSED DICTIONARY	Spell 'n Fix uses either of two forms of dictionary, and converts either to the other at your command. The first type is a fully expanded dictionary with one word per line, allowing you to use your editor to make corrections, deletions and additions easily. The second type incorporates a slick compression method to drastically reduce dictionary size and increase access speed.

After looking at our options, we decided that this was the program for us, and are proud to be the exclusive agents for StarKits for the SSB DOS68 and DOS69 versions. We can also supply versions for 6800 and 6809 Flex. The OS-9 version will be available soon directly from StarKits. StarKits also has a version for the Color Computer! At the price, we feel that no other program can do what Spell 'n Fix can....

SPELL 'N FIX disk (with 17000 word dictionary) and manual $89.29
(manual not available separately)

SS50 SPEECH SYNTHESIZER

The VS-1 is a completely-assembled, tested unit for the SS-50 I/O bus which gives your computer the power of unlimited speech. Since the VS-1 is programmed phonetically, there is essentially no limitation on what you can make your computer say! The VS-1 programs just like a parallel printer driver, and requires an average of only one phoneme per letter in English for a very low message memory storage overhead. The data rate is typically a low 70 baud.

The VS-1 is supplied with the easiest-to-use, most flexible speech editing program available. This program makes building your speech files or data a real pleasure. We also supply additional software and sample speech files on the disk to give you a 'leg up' on making your computer talk.

The manual provides sections on speech generation theory, operation of the supplied software, complete technical descriptions, a sample speech dictionary and phonetic code charts, fifty pages in all!

If you are a roll-your-own nut, we provide the bare board or manual or disk or synthesizer chip, or combinations (We suggest that only VERY experienced persons attempt such assembly due to the fact that we do not provide assembly instructions beyond parts list parts layout and schematic).

The VS-1 is available for 6800 and 6809 FLEX, 6800 and 6809 SSB DOS, and OS-9 level I and II.

VS-1 complete with manual and disk $194.95
VS-1 manual and disk only (may be adapted for brand-X) . . . . . $ 29.95
VS-1 manual only (may be applied to VS-1 order. Look before you leap!) . . . . $ 19.95
VS-1 bare board (build your own...requires experience!) . . . . . $ 19.95
SC-01A synthesizer chip . . . . $ 60.00
VS-1 bare board, chip, disk and manual . . . . $ 99.95

TRANSLATOR FOR VS1

If you have our VS-1 (or even brand-X), and want an easier way to make your computer talk, our SC-1 translator is for you. The SC-1 takes plain English text from a disk file or your keyboard, translates it to a phoneme stream, and sends the resulting output to either the speech synthesizer hardware or to a data file for later use.

The SC-1 translator is one of the most accurate such programs we have seen, making typically only one 'mistake' per sentence, for an average accuracy of about 95%! Due to the type of algorithm we are using, even proper names are generally translated correctly! Where the SC-1 is being used to compile 'canned' messages for later use, we provide an additional program which takes a binary phoneme file created by the SC-1 and converts it to a speech file which may be edited by our VOX-EDITOR (supplied with the VS-1). In this way, you can quickly create your messages and then 'fine-tune' them!

The SC-1 translator is available for 6800 and 6809 FLEX, 6800 and 6809 SSB DOS, and OS-9 level I and II. If you are using brand-X, order the SC-1 with the VOX-EDITOR on the same disk!

SC-1 disk and instructions $39.95
SC-1 disk, instructions with VOX-EDITOR $49.95

TREK-68

Our TREK program is the best 'real-time', 'live-action' game available for 68xx computers using memory-mapped displays. This program does not require graphics capabilities from your display. Otherwise, this is a full arcade-type game. No triviality here, either. The suicide option has only been won by a handful of people world-wide, and can run for hours!

TREK is available for 6800 and 6809 FLEX and 6800 and 6809 SSB DOS, and requires both a memory-mapped display and a source of 60 or 100 Hz interrupts (such as the MP-T timer board).

TREK disk and manual $24.95

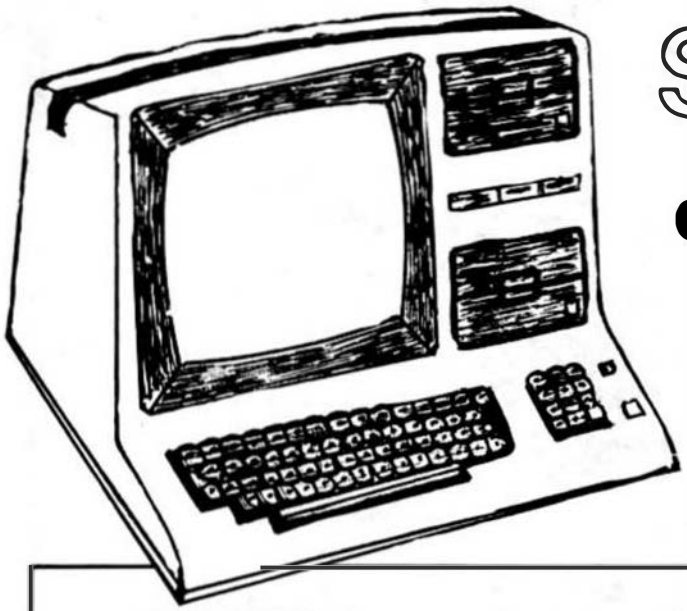
COLOR-TREK

Like our other version of TREK, COLOR-TREK provides you seat-of-the-pants action. This program uses no routines from Basic, and so can run on non-extended Basic machines. 16k memory is required. Only one player to our knowledge (other than the author) has won the suicide option of this game, and it took him fifty-seven minutes! No triviality here!

COLOR-TREK cassette and manual $17.95

ALFORD & ASSOCIATES
P.O. BOX 6683
RICHMOND, VA., 23230
804-320-6722

Universal Data Research, Inc. Introduces



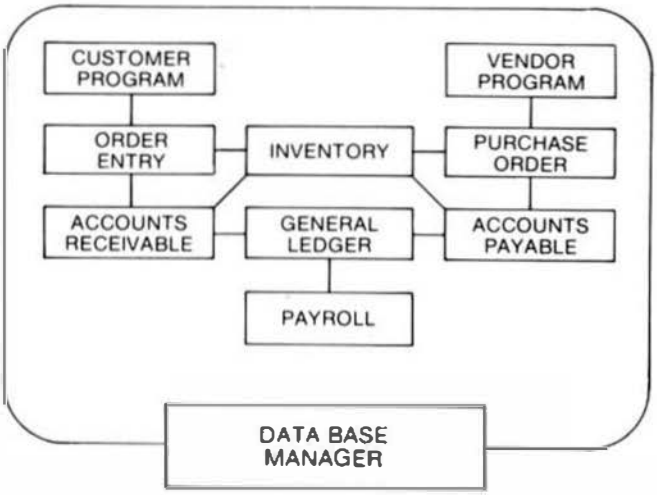
SOFTWARE
for the COLOR COMPUTER and the TRS 80 Model III

- Data Base Manager _____ $150
- Church Contributions Package _____ $150
- Single Entry Ledger _____ $ 95

FLEX* and UniFLEX* SOFTWARE
for the 68XX Operating System . . .

INTEGRATED BUSINESS PROGRAMS
for large and small companies

	FLEX	UniFLEX
Accounts Receivable	$395	$495
Accounts Payable	$395	$495
General Ledger	$395	$495
Inventory 2	$395	$495
Payroll	$395	$495
Data Base Manager	$450	$550



STAND-ALONE SOFTWARE

- Single Entry Ledger
- Check Balancer
- Costing and Estimating
- Church Contribution Pkg.
- Church School Billing
- Fund Raising
- Custom Programming
- M.F.I. (Extensive Mfg./P.O.)
- Bulk Mailings
- Maintenance Schedule
- Vendor Program
- Purchase Order
- Accounts Payable
- Accounts Receivable
- Customer Program
- Order Entry

UNIVERSAL DATA RESEARCH INC.

*FLEX & UniFLEX are Trademarks of Technical Systems

2457 Wehrle Drive,D-1, Buffalo, New York 14221
Phone (716) 631-3011

VISA

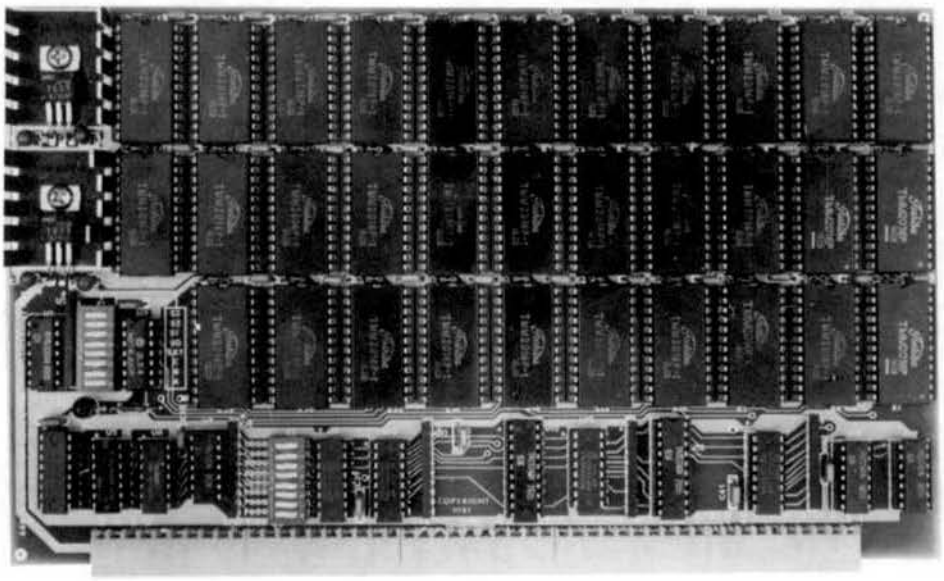
64K SS-50 STATIC RAM

NEW!

$219.00 (48K KIT)

NEW!

LOW POWER!



RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION $52

SUPPORT ICs + CAPS -	$18.00
FULL SOCKET SET -	$15.00

ASSEMBLED AND TESTED **ADD $40**

FEATURES:

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- ★ 2716 EPROMs may be installed anywhere on Board.
- ★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- ★ One Board supports both RAM and EPROM.
- ★ RAM supports 2MHZ operation at no extra charge!
- ★ Board may be partially populated in 16K increments.

56K Kit	$269
64K Kit	$319

16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power, RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

Digital Research Computers
(OF TEXAS)
P.O. BOX 401565 • GARLAND, TEXAS 75040 • (214) 271-3538

TERMS: Add $2.00 postage. We pay balance. Order under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P. & H. Orders over $50, add 85¢ for insurance.

ARCADE 50

POWERFUL COLOR GRAPHICS
Uses the new TMS9918A Video Display Processor. High resolution 256 x 192 pixel display with 15 colors.16k Bytes of onboard RAM does not reduce user memory. 32 graphic images can be individually moved with simple X-Y commands for smooth animation.
External Video input allows subtitling
NTSC composite video output
SOUND EFFECTS AND MUSIC
Three AY3-8910 Programmable Sound Generators
Nine simultaneous voices
Three independent noise sources
Onboard stereo amplifier drives two 8 ohm speakers
ADDITIONAL I/O CAPABILITIES
Eight analog inputs with 8 bit resolution
Supports four joysticks with pushbutton switches
Eight bit parallel I/O port
Entire unit maps into 256 bytes of memory
DOCUMENTATION AND SOFTWARE
Programming manuals for Video and Sound Processors
Subroutine library and Super Demo Maze Game
Example programs in BASIC, FBASIC and ASSEMBLY
User library and sales support

ARCADE 50,assembled and tested	$325.00
Video and Audio connector set	15.00
4 Joystick connector set	15.00
2 Radio Shack Joysticks	24.00
UHF channel 33 modulator	32.00
Gold Molex connectors	12.00
A/BASIC for 6800	110.00
FBASIC for 6809	110.00
FBASIC (with ARCADE 50)	75.00
FBASIC (manual only)	10.00
ARCADE 80 (TRS Model I)	395.00
ARCADE 100 (S-100 BUSS)	375.00
ARCADE 50 RGB	375.00
LABVIDEO (Motorola EXORbus)	375.00
LABVIDEO RGB	375.00
NEW MV09 6809 Processor Board	225.00
*Comes assembled with PIA and ACIA	
*12 Sockets for 2716, 2732 or RAM	
*Supports DMA disk I/O	
*Ideal for 6809 upgrade or process control	
AMDEK COLOR I Monitor	425.00
AMDEK COLOR II Monitor	799.00
AMDEK COLOR III Monitor	499.00
256K Dynamic Memory Board (assembled)	795.00
256K Dynamic Memory Board (assembled w/64K)	395.00
64K Dynamic Memory Board (assembled)	295.00

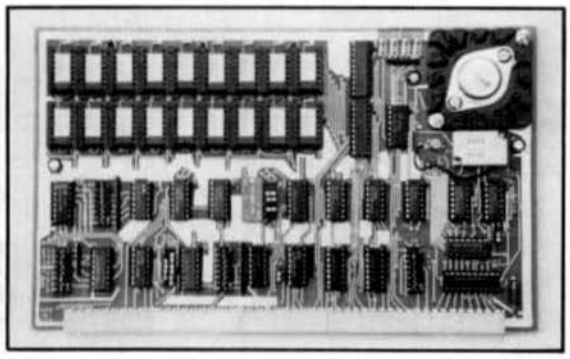
Specify 5" or 8" soft sector disk for TSC's FLEX or MICROWARE'S OS/9 system.
TERMS: CASH, VISA, MC, C.O.D.

FBASIC

TERMINUS DESIGN INC., in conjunction with Microware Systems Corporation, is proud to announce FBASIC-an enhancement of Microware's 6800 A/BASIC. Their fast compiled BASIC has been adapted for 6809 users with added video and sound features for ARCADE 50 users. FBASIC is a true compiler that produces optimized machine language modules which are ROMable and require no Run-Time package. FBASIC requires less memory overhead and runs hundreds of times faster than BASIC interpreters. It supports standard BASIC instruction including String functions, Disk I/O and fast integer arithmetic with multiple-precision capability. Graphics verbs and functions fully support the Arcade 50. Arcade statements include:

INIT	MODE	BLANK	BACKDROP
SIZE	MAG	VREG	DELAY
MOVE	DRAW	FCOLOR	JSWITCH
REMOVE	RDRAW	BCOLOR	SWITCH
PSG	TONE	ENVL	VOLUME
ADC	SPRITE	SPNAME	ENDEF
SPCOLOR	RSPRITE	SPDEF	PATDEF
VPEEK	VPOKE	VPRINT	

TERMINUS DESIGN INC
16 SCARBROUGH ROAD
ELLENWOOD, GA 30049
(404) 474-4865



QMM1-B 256K MEMORY FOR SS50-C 6809 SYSTEMS

Compatible with systems by SSB, GIMIX, and SWTPC including those with DMA disk controllers.

Full 2Mhz operation with transparent on board refresh, runs continuously at 2 Mhz without cycle stretching or stealing.

Very versatile addressing and disable features.

Parity option halts processor and sounds audible alarm upon detecting a read error.

All boards assembled, tested, burned in and warranted for 1 year.

Also available with 64K, 128K, or 192K.

256K for $1135.00 — 256K w/ parity $1270.00

Delivery: Stock—2 weeks. Terms: Prepaid or COD.

D.P. Johnson (503) 244-6152
7655 S.W. Cedarcrest St., Portland, OR 97223

AUTO-COMM

NEW!

For 6809 FLEX

The 'modem' program that automates time sharing communications.

FEATURES:

1) Send text file from personal computer disk to remote mainframe computer.
2) Save incoming text to disk file (verifys acceptance of XON/XOFF controls).
3) Slow transmission mode based on character verify for systems which require speed below baud rate.
4) Eight software selectable UART modes; 8 bit, 7 bit.
5) Self adapts to amount of memory in your computer.
6) Runs in as little as 12K bytes or up to 65K bytes.
7) Reads and writes files of phone numbers to be dialed.
8) Makes any modem a smart modem.

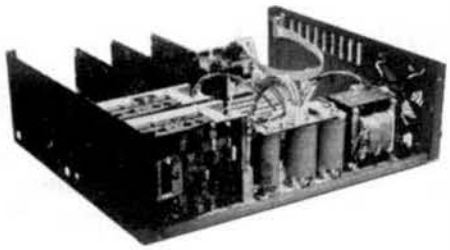
Call Or Write For Further Information
Priced at $75 Manual only $3.95

SYSTEMS designware

6712 EAST PRESCOTT ST. • SCOTTSDALE, AZ.85254
VISA-MASTERCARD ACCEPTED
ORDER TOLL FREE 800 272-4817 (IN AZ.) 991-1657
FOR INFORMATION CALL (602) 991-1657

AAA Chicago Computer Center

ELEKTRA COMPUTER PRODUCTS



ELEKTRA CABINET Made of heavyweight 0.090" thick aluminum. Interior is 18-1/2" wide by 21-7/8" deep by 6-3/4" high. Heavy duty A.C. line cord, A.C. fuse holder, EMI filter. Fan with filter. Back panel has 10 cutouts for D type data connectors. Front panel has key on/off power switch, 2 illuminated push button switches (Reset and NMI/Abort), and two cutouts for 5-1/4" disk drives.

POWER SUPPLY Highest quality linear power supply CONSERVATIVELY rated at 15a @ 8v, 3a @ 16v, 3a @ -16v, 3 primary inputs for light rated, and heavy loading.

DISK REGULATOR BOARD WITH CABLES Standard version for 2 floppy drives. Heavy duty version for 1 Winchester drive and 1 floppy drive.

ELEKTRA MOTHERBOARD Heavyweight 0.125" thick, 18" long by 9" wide. 11 memory (50 pin) slots, 4 or 8 slots may be cut off for shortening to 14" or 10" lengths respectively. 8 I/O (30 pin) slots. Complete address decoding of I/O slots. Choice of 4, 8, or 16 addresses per I/O slot. Base addresses for I/O slots can be placed at 32, 64, or 128 byte increments respectively. 1" spacing between all memory and I/O slots. Extended addressing capability for both memory and I/O ports for meeting SS-50C bus specifications. On board baud rate generator with low and high ranges providing jumper selectable rates of 75 through 38,400 for each of the five baud rate lines. Slow device circuitry permitting 1MHz 30 pin disk controllers to run with 2 MHz 50 pin CPU boards.

ELECTRA CPU 8/9 Use either the 6802 or 6808 (to run 6800 software) or 6809. Has provision for up to 3 2716 Eproms, 1K scratchpad, MC6840 triple timer, and an optional baud rate generator providing baud rates from 110 through 38,400 baud in two user selectable ranges. The board supports DMA by either HALT or BUSREQ when a 6809 CPU is used. The board does not support a DAT and therefore does not support extended addressing. The board will run any of the MIKBUG™ compatible monitors in the 6802/6808 mode and SBUG-E, HUMBUG, and GMXBUG-09 in the 6809 mode. The ELEKTRA CPU 8/9 will run any of the popular disk controller boards with the appropriate software. OS-9™ Level 1 is available as shown below.

ELEKTRA DPS DUAL PORT SERIAL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 addresses per port with the B port 2 addresses higher than the A port or for 16 addresses per port with the B port 4 addresses higher than the A port. Each port is terminated at two 16 pin dip sockets, one socket configured for modem and the other for terminal or printer. RTS, CTS, DTR, DCD, and DTR are appropriately implemented. Each port has independent selection of baud rate. Each port allows the interrupt request to be independently jumpered to the IRQ or FIRQ/NMI bus line.

ELEKTRA DPP DUAL PORT PARALLEL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be used in either the 4 or 16 addresses per I/O slot configuration occupying the first four addresses of the I/O slot. The direction of the TTL buffers can be controlled by either on board jumper connectors or by a signal from the peripherals. The interrupt request lines for each port may be individually jumpered to either the IRQ or FIRQ/NMI bus line.

ELEKTRA CHASSIS Includes cabinet, 110v power supply, power supply cables, standard disk regulator board with power cables, assembled motherboard with gold square pin connectors $825.00*

	SSB DOS	FLEX™	OS-9™ L1
ELEKTRA CHASSIS with 2 GIMIX 32K memory boards (64K to 1)			1300.00*
ELEKTRA CHASSIS, 2 32K GIMIX memory boards, ELEKTRA 8/9 CPU			1550.00*
The $1550.00 combination above with disk controller board and software as indicated (monitor must be added)			
SSB DCB-4A (SD/DD, controls up to 8 floppies)	2095.00	2245.00***	2295.00***
***Includes SSB DOS also			
GIMIX #68 DMA Controller	N/A	2225.00*	N/A
GIMIX #28 Double Density, PIO Controller	N/A	1935.00*	N/A
GIMIX #58 Single Density, PIO Controller	N/A	1865.00*	1975.00*

Other combinations available

ELEKTRA CABINET	260.00
ELEKTRA 110v POWER SUPPLY	175.00*
ELEKTRA CABINET, 110v POWERSUPPLY, AND POWER SUPPLY CABLES	410.00*
ELEKTRA CABINET, 110v PS, PS CABLES, STD. DISK REGULATOR AND CABLES	460.00*
*Add $30.00 for 220v	
STANDARD DISK REGULATOR BOARD AND CABLES	50.00
HEAVY DUTY DISK REGULATOR BOARD AND CABLES	75.00
FILLER PLATE FOR 5-1/4" DRIVE OPENING	10.00
FAN FILTER	10.00

See our ad on the next page for our ELEKTRA floppy drive cabinets.

GOLD 10 PIN CONNECTORS (Specify male w/square pins or female)	1.50
TIN 10 PIN CONNECTORS (Specify male w/square pins or female)	.50

	BARE BOARD	KIT	ASSEMBLED
ELEKTRA MOTHERBOARD (Gold connectors)	80.00	320.00	380.00
ELEKTRA MOTHERBOARD (Tin connectors)		240.00	300.00
ELEKTRA CPU 8/9	50.00	225.00	275.00
ELEKTRA CPU 8/9 with baud rate option		250.00	300.00
ELEKTRA DPP DUAL PORT PARALLEL BOARD**	20.00	60.00	80.00
ELEKTRA DPS DUAL PORT SERIAL BOARD**	20.00	60.00	80.00
**CABLE FOR DPP OR DPS (2 needed, specify board)			20.00

MODEMS	MANUAL ANSWER	AUTO ANSWER
1200 Baud (120 cps), direct connect (U.S. Robotics)	449.00	549.00
300 Baud (30 cps), direct connect (U.S. Robotics)	179.00	219.00
300 Baud (30 cps), acoustic (U.S. Robotics)	149.00	N/A
1200 Baud (120 cps), direct connect (Hayes Smart Modem)		595.00
300 Baud (30 cps), direct connect (Hayes Smart Modem)		229.00

WARNING AAA Chicago Computer Center does not provide repair or diagnostic service for customer assembled kits. AAA Chicago Computer Center does not warranty and maintain service for our assembled boards. The customer should carefully take into consideration the small differential separating our kit and assembled prices when making his choice of purchase.

We have introduced our line of computer equipment with the purpose of offering the highest quality of components possible at affordable prices. These products are intended for OEM applications where it is the responsibility of the purchaser to integrate these components with suitable memory, disk controllers, drives, and software along with I/O terminals to form working computer systems.

Prices and inventory are subject to change without advance notice.
This ad is our catalog.

SUPER MODEM PROGRAM

Transmit manually to distant computer.

Transmit disk files (text) of any length to distant computer.

Receive and save disk files (text) of any length on disk system. If sending computer does not support an X-on/X-off protocol, then the received files are limited in size by the computer memory.

Tested to transmit and receive text at speeds up to 9600 baud. (CRT terminal must be capable of operating at a baud rate higher than the one the modem is operated at.)

Half duplex option in case distant computer doesn't echo.

Echo option so user can simulate a time sharing system. (Super Modem Program doesn't support auto-answer but the source is provided for those individuals who wish to adapt our program to their special needs.)

Replaces CR with CR/LF (user option) for those using time sharing systems that don't transmit LF's.

Slow disk file transmit (user option) based on character verification for use on time sharing systems to which disk files can not be sent at speed suggested by the baud rate.

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8".
Manual and disk with both source and object code $75.00

STANDARD MODEM PROGRAM

Same as Super Modem Program above but without ECHO option, CR/LF for CR option, slow disk file transmit option, nor X-on/X-off option. Reception of disk files is limited to those small enough to completely fit within the receiving buffer.

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8".
Manual with instructions, source listing, and flow chart, disk with both source and object code $45.00
Manual with instructions, source listing, and flow chart 25.00

TERMS Minimum order $20.00. Shipping and handling estimates within the Continental U.S., add 3% (MINIMUM $2.50). Illinois residents add 6% sales tax. We will refund your overestimated shipping and handling charges. Foreign shipping and handling, add 10% (MINIMUM $10.00). Foreign orders must be prepaid in U.S. dollars. Heavy foreign items will be shipped air freight collect. Please phone between 4 PM and 8 PM weekdays if questions arise regarding shipping fees. Master Charge, Visa, and American Express honored.

Our apology. We are not staffed to answer technical inquiries through the mail. Please phone for technical help during the hours indicated above. The too frequent changing of our inventory and prices makes it uneconomical to publish a catalog. Our ads are intended to serve that purpose. Prices and inventory are subject to change without advance notice.

AAA CHICAGO COMPUTER CENTER
120 CHESTNUT LANE • WHEELING, IL 60090
(312) 459-0450
Technical consultation available 4 PM to 6 PM most weekdays. Closed evenings and weekends.

SMOOTH™ Software

ALL IN ONE

Editor - Text Processor - Mailing Labels
Mailing Lists - Use any CRT terminal and printer

Supports Editing commands such as bottom, change, delete, find, insert (single line), input (multiple lines), list, next, overlay (with cursor editing, character deletion and insertion), overstrike (for selected darker text), Print, restart, set, top, underline, up and verify.

Supports Text Processing commands such as block copy, block move, centering, margin justification (widen and narrow), paging, and tabbing.

Mailing Lists and Labels. Use the same mailing list disk file (with protected areas) for both mailing labels and repeat letters. Repeat letters are personally adressed to each person or selected persons on the mailing list.

Most Powerful File Handler found in any editor. Append one file to the end of another, or insert (merge) one file into another as designated by the line pointer. Print specified lines to your printer or to a disk file. Edit files larger than the text buffer. Does not produce output files when not desired. Delete disk files from the editor.

Printer commands. Control characters can be sent to the printer for formal control either directly from the control terminal or by imbedding them in the text. The set command contains interface initialization and character output routines to support the SWTPC MP-C interface as well as the standard serial and parallel interfaces. Jumps are also provided to user supplied printer routines. User selects the port address (0 thru 7, A or B) thereby eliminating the need for the user to install printer software routines. Editor can be initialized for either 4 or 16 addresses per port.

Editor allows exiting to either the monitor or DOS and then reenter (Warm Start) without destroying previously prepared text in the buffer. The Restart command erases contents in the buffer without the user having to reload the Editor.

The Editor allows the user to toggle between full duplex (no echo) and half duplex (echo) as needed. It responds to commands in both upper and lower case and can be used to create assembler source code and Basic programs as well as text.

Specify 6800 or 6809, SSB or FLEX™, 5" or 8"	75 00
Printed source listing is available for an additional	35 00
All-in-One, Write'n Spell, and Spell'n Fix package	195 00

Software by Technical Systems Consultants, Inc.

Flex™ (includes Editor and Assembler)	150 00
UniFLEX™ (includes one year maintenance and update)	550 00
Editor	50 00
Assembler	50 00
68000 Cross Assembler on 6809	250 00
6809 Cross Assembler on 6800	100 00
Text Processor	75 00
Extended Basic	100 00
Basic Precompiler (specify standard or extended)	50 00
Basic for UniFLEX™ (includes one year maintenance and update)	200 00
Pascal (Flex™)	200 00
Pascal (UniFLEX™) (includes one year maintenance and update)	300 00
Sort-Merge Package	75 00
6809 Flex™ Utilities	75 00
6800 Flex™ Utilities	100 00
Debug Package	75 00
Diagnostic Package	75 00

Software by Microware Systems Corp.	UPDATE	SOURCE	MANUAL	OBJECT
OS-9™ Level One Operating System	75 00	400 00	40 00	200 00
OS-9™ Level Two Operating System	75 00	N/A	40 00	500 00
BASIC09™	75 00	N/A	25 00	200 00
BASIC09™ Run-Time Package				100.00
OS-9™ Macro Text Editor		300 00	15 00	125.00
OS-9™ Interactive Assembler		300 00	10 00	125.00
OS-9™ Interactive Debugger (Disk version)		100 00	10 00	50.00
CIS Cobol Compiler	250 00	N/A	80 00	900 00
CIS Cobol Run-Time Package				400 00
Pascal Compiler	100 00	N/A	40 00	400 00
Pascal Run-Time Package				100 00
Microware yearly support service ($200 00 for OS-9 Level 2)				75 00

SWTPC	Kit	Assembled
DMF2C Disk Controller Board		N/A
6809 SWTPC FLEX™ Disk and manual		35.00
6809 SWTPC FLEX™ Disk without manual		15 00
DC-4 Disk Controller	N/A	230 00
SBUG-E (2716 compatible)		25.00
MP-A2 6800 CPU BOARD		150.00
MP-S Serial interface (single port, limited quantity)	0 00	60 00
MP-S2 Serial interface (dual port)	N/A	120 00
MP-LA Parallel interface (dual port, limited quantity)	N/A	N/A
MP-L2 Parallel interface (dual port)	N/A	120 00
MP-R Single voltage 2716 prom programmer	N/A	114 50
MP-N Calculator board	54 95	92 00
MP-T Interrupt timer	N/A	92 00
S32 Universal Static Memory Board	N/A	124 50
MP-09 6809 CPU board (Used $225 00)	N/A	295.00
69A Chassis, P.S., 68809 CPU, 8K, RAM, One Serial Port	N/A	N/A

Universal 68XX MBT Bare Motherboard, 6800/6809, 4/16 addresses per port, 8 50 pin/8 30 pin slots, baud rate generator, 6840 interrupt timer, slow device circuitry, 15 1/8" x 9 3/8" — 55 00

Connectors (10 pin)	Tin	Gold
Male with square pins	0 50	1 50
Female	0 50	1 50

Special Software

4K 6809 HUMBUG	75 00
4K 6800 HUMBUG (RAM needed at $A000 and $D000)	65 00
2K 6800 HUMBUG (With cassette LOAD and PUNCH)	0 00
2K 6800 HUMBUG (Extra commands instead of cassette software)	40 00
Other HUMBUG versions including video versions are available	
Spell'n Fix by Peter Stark	89 29
Write'n Spell by Peter Stark	75 11
All-In-One, Spell'n Fix, and Write'n Spell package	195 00
Dynamite Disassembler	60 00
SUPER SLEUTH Disassembler System ($101 00 for OS-9 version)	99 00

DISK DRIVES 30 day guarantee, SD/DD capability	1 head	flippy	2 heads
5-1/4", 40 tracks	250 00	260 00	325 00
5-1/4", 80 tracks	325 00	335 00	425 00
MPI — Service Manual (General, covers 40 and 80 track)			20 00
MPI — Service Manual (Specify 40 or 80 track)			25 00
Siemens Manual			10 00
8", 77 tracks	375 00		495 00
8", 77 tracks, Thin-Line	450 00		525 00

Microbyte II Calendar and Clock Board	60 00
Data Mart 16K EPROM bareboard (2708 chips)	30 00

SMOOTH™ and **ELEKTRA™** are trademarks of AAA Chicago Computer Center
FLEX and **UniFLEX** are trademarks of Technical Systems Consultants, Inc.
OS-9 and **BASIC09** are trademarks of Microware Systems Corp.
GIMEX* and **GHOST*** are registered trademarks of GIMIX Inc.

ELEKTRA™ SS50 Computer Products

ELEKTRA D-5 Dual drive cabinet for 5-1/4" drives with power supply, line cord, fuse, power switch, and power cable to drives	125 00
ELEKTRA HD-5 (Heavy duty version of D-5 package above)	150 00
ELEKTRA SHD-5 (Super heavy duty. Powers 1 Winchester and 1 floppy)	175 00
5' ribbon cable for dual 5 1/4" disk drives	40 00
ELEKTRA D-8 Dual drive cabinet, power supply, ps cable for 8" drives	350.00
Cabinet for dual 8" drives only	250.00
Power supply for dual 8" drives only	120.00
PS cables only (Specify brand and type of 8" drives)	30.00
6' ribbon cable for dual 8" disk drives	45.00

TERMINALS

Hazeltine 1420	495 00
Hazeltine Espirit 1	525 00
Adds Viewpoint (Green Screen)	525 00
Televideo 925 (Green Screen)	775 00
Televideo 950 (Green Screen)	995 00

Printers

Okidata ML 82A (120 cps, 9x9, bidirectional, serial and parallel)		475 00
Tractor for ML 82A		60 00
Okidata ML 83A (120 cps with tractor)		775 00
2K byte buffer (High speed RS-232)		125 00
Dot addressable graphics		50 00
Okidata ML 84 (200 cps, 2K, Graphics, Parallel)		1195 00
Okidata ML 84 (200 cps, 2K, Graphics, Serial)		1295 00
TI 810 w/lower case and full vertical forms control (limited Quantity)		1400 00
Florida Data (600 cps)		485 00
NEC 3510 Std		1495 00
NEC 7710 Std		2445 00
Epson MX-80FT (Centronics compatible, parallel interface)		525 00
(with Serial RS-232 interface option)	add	75 00
Spare Print Head		39 95
Spare ribbon cartridge		9 50
Epson MX-100		725 00
C. Itoh Prowriter 8510 (Parallel Interface, 120 cps)		525 00

Optimal Technology, Inc. EP-2A-79 Eprom Programmer (Personality Modules extra for above programmer)	168 00
Optimal Technology, Inc. 30 pin parallel I/O board for EP-2A-79	37.00
Software package for EP-2A-79 (Specify 6800 or 6809)	30 00

Smoke Signal Broadcasting

DCB-4A Double Density Controller Board for 5" and 8" with DOS	549 00
SSB DOS Specify 6800 or 6809, BFD or DCB-4A, 5" or 8"	75 00
SSB Monitor (Specify 6800/6809, $8000/$E000/$F7E0)	75 00
SSB version of FLEX™ (without Editor and Assembler)	150 00
LMB-1A Motherboard	399 00
SCB-69 6809 CPU Board	399 00
PAR-1 Dual Port Parallel Board	89 00
SER-2 Dual Port Serial Board with 2 Cables	129 00

Static Memory Boards	M-16-X 195 00	M-24-X 295.00	M-32-X 395 00
Dynamic Memory Boards	M-128-X 995 00	M-256-X 1295.00	M-512-X 1895 00

GIMIX

6800 CPU Board		224 03
with fillers		288 06
with baud rate option	add	30 00
with 2MHz option	add	15 00
2 MHz 6809 Plus CPU, time of day clock, battery backup, 1K NMOS RAM		578 05
CMOS RAM substitution		8 00
GIMIX Dynamic Address Translator		35 00
SWPTC compatible DAT		15 00
9511 A Arithmetic Processor (4MHz)		312 00
9512 Arithmetic Processor (1 MHz)		265 00
GMXBUG-09 (Terminal Based) 1K scratchpad required		98 65
Manual and Source Listing only		38 62
Bootstrap Prom		30 00
Video Prom (includes bootstrap)		30 00
Filler plate for 5-1/4" drive opening		14 92
Baud rate generator board		88 93
Missing cycle detect card		38 23
Prototyping board (50 pin slot)		58 86
Prototyping board (30 pin slot)		38 33
Disk Controllers (All have data separators and can be used with either single or double headed drives)		
5" single density controller without 1771 chip		158 38
5" single density controller complete		198 48
5" and 8" single density controller complete		228 58
5" double density controller with variable precomp		298 28
DMA 5" AND 8" double density controller with variable precomp		588 68
GIMIX version of FLEX™ (without Editor and Assembler)		90 00
Double disk regulator card		88 22
Ribbon cable for two 5 1/4" disk drives (inboard)		34 98
Ribbon cable for two 8" disk drives (outboard)		44 28
8" disk ribbon cable and back panel connector set		29 25
8" disk drive cabinet with power supply		848 18
19MB Winchester and controller update		4288 90
38MB Winchester and controller update		6888 91

Memory	CMOS WITH BAT BACKUP	NMOS NO BAT BACKUP
64K Static RAM Board with 24K of RAM installed	N/A	348 27
64K Static RAM Board with 32K of RAM installed	518 36	398 37
64K Static RAM Board with 48K of RAM installed	N/A	518 47
64K Static RAM Board with 56K of RAM installed	728 56	578 57
64K Static RAM Board with 64K of RAM installed	798 64	638 67
16 Socket EPROM/ROM/RAM Board		238 32
8K Promboard (2708)		98 34
32K Static RAM Board with 32K of RAM installed		245.00

I/O Boards	1 port	2 port	8 port
Serial interface	88 41	128.43	318.46
Parallel interface		88.42	198.45
Cable sets for above boards (specify board)		each	22.95

Video Boards	
64 or 32 X 16	198.71
80 X 24 without RAM character generator	398.74
80 X 24 with RAM character generator	458.76
High resolution (512 X 512 dot resolution)	996.77

2MHz 6809 PLUS Computer System	w/o Disk Cont	w/#58 Cont	w/#68 Cont
With 56K of memory	2498 29	2988 59*	3248 49*
With 128K of memory			3798 39
With 128K and 19MB Winchester			[illegible] 09
With 128K, 38MB Winch, dual 8"			17498 99

*Includes GMXBUG/FLEX/OS-9 software selectable
*With CMOS RAM and Battery Backup — add 150.00
Mainframe (Chassis, PS, Switches, Fan, Motherboard, Baud Rate Gen.) — 1198.19

AAA Chicago Computer Center

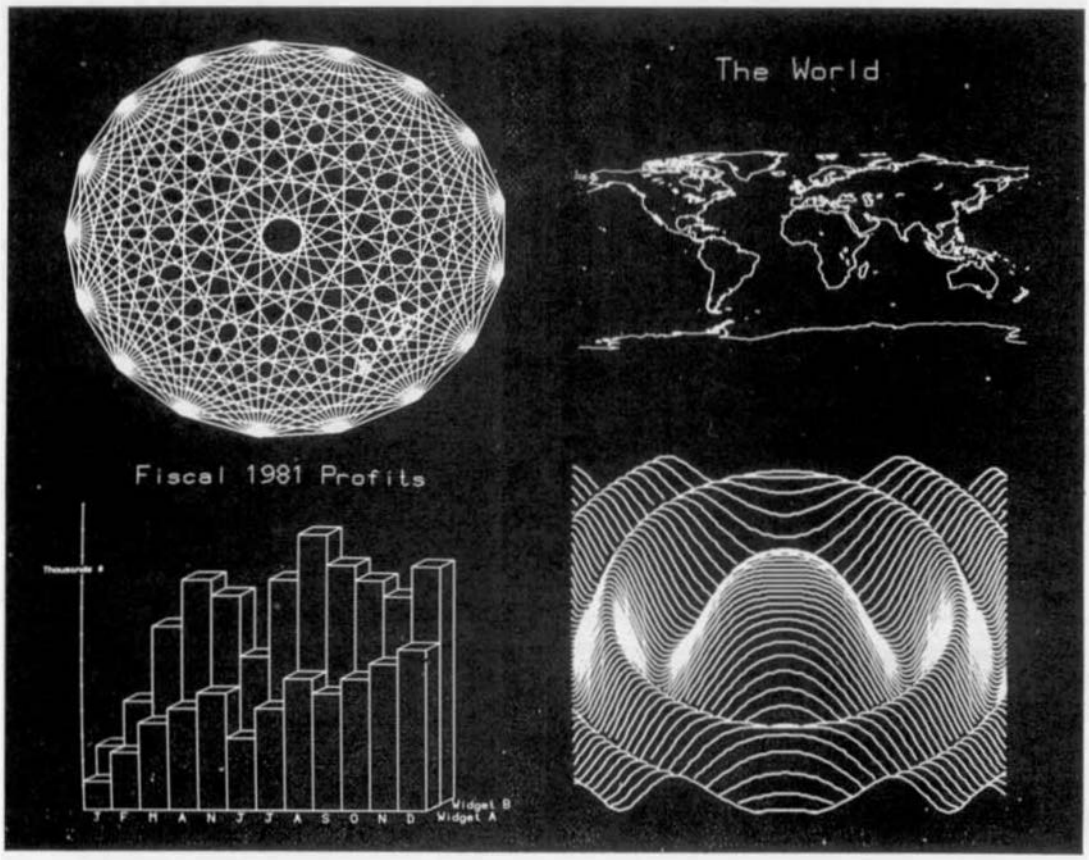
See our ad on the previous page to your left for ordering instructions.

ANNOUNCING ElectroScreen™

only $595

the Superior Alternative

to the Traditional Alphanumeric Terminals



The ElectroScreen™ Intelligent Graphics Board Features:

Graphics

- 512 x 480 resolution bit-mapped display
- Interleaved memory access — fast, snow-free updates

Intelligence

- 6809 on-board mpu
- 6K on-board firmware
- STD syntax high level graphics command set
- Removes host graphics software burden
- Flexible text and graphics integration
- Multiple character sizes
- User programs can be run on-board

Terminal

- Terminal emulation on power-up
- 83 characters by 48 lines display
- Easy switching among user-defined character sets
- Fast hardware scrolling

Additional Features

- SS-50C and SS-64 compatible board
- Board communicates with host through parallel latches
- Composite and TTL level video output
- 8 channel 8 bit A/D converter
- Board occupies 4 address bytes

See your dealer today!

The ElectroScreen manual is available for $10. credited toward purchase of the board.

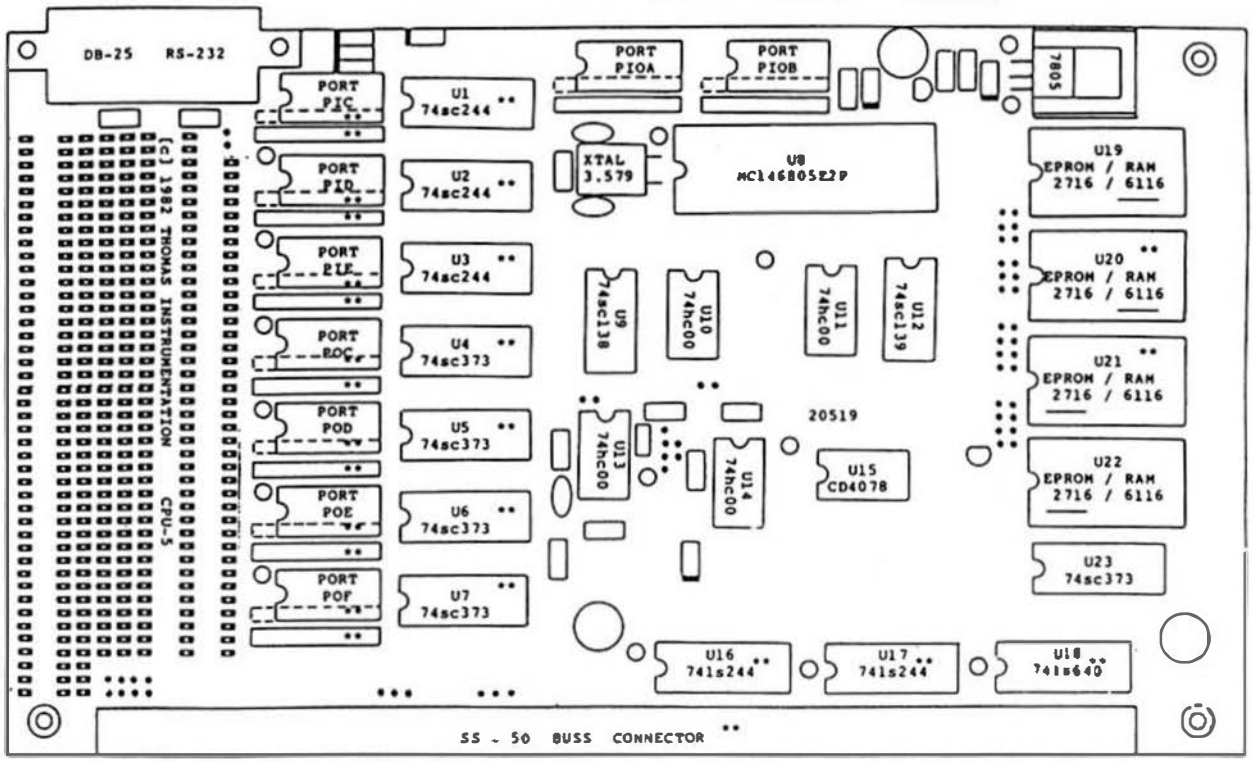
The ElectroScreen has a 90 day warranty from purchase date.

Dealers, please contact us for our special introductory package.

Privac Inc (703) 671-3900

3711 S. George Mason Dr., Falls Church, Va. 22041

*** NEW 6805 ALL CMOS CARD ***



CPU-5-B	Blank card with documentation	$49.00
CPU-5-P	Partial card (parts marked ** not incl.)	$229.00 } Includes EPROM Monitor
CPU-5-F	Full card with all parts shown (fully assembled)	$349.00 } and 115 vac Power Pak.
CROSS-ASSEMBLERS	for FLEX ((c) TSC) 6809 system	$150.00

SOFTWARE

VDISK Treat extended memory as a super fast disk drive

6809 source & object	$149.00
6809 object	$ 99.00

OUTSIDE MODEM PROGRAM incl. source

UniFLEX version	$100.00
FLEX version 6800 & 6809	$ 50.00
HAYES SMART MODEM	$249.00

CROSS ASSEMBLERS for 6800,6801,6805 runs on 6809 FLEX $150.00

TV-EDIT Screen oriented Editor for FLEX 6809 $ 95.00

Please specify 5" or 8" Disk when placing order.

CHIPS

Please note when ordering IC's that a $100.00 Minimum is in effect Orders less than the minimum will be charged an additional $10.00 for handling

6116P-3	2MHZ	$13.50
2016P-2	2MHZ	$10.50
2716	1MHZ	$ 6.50
2716-1	2MHZ	$16.50
146805E2P		$25.00
68B02		$10.00
68B09		$25.00
68B21		$ 6.50
68B40		$12.00
68B50		$ 6.75
74LS640		$ 3.25

HARDWARE

NEW MEMORY PRICES TO INTRODUCE VDISK

S-R/R without memory chips	$120.00
with 8K NMOS@2MHZ	$169.00
with 16K NMOS@2MHZ	$199.00
with 32K NMOS@2MHZ	$299.00
with 48K NMOS@2MHZ	$399.00
Bare Card	$ 49.00

Extender Cards assemb. with logic ald

SS-50/50C	$ 35.00
SS-30	$ 25.00
SP-1 Prototype Board A/T	$195.00
SP-1 Bare Card	$ 49.00
SS-50 Wire Wrap Board B/C	$ 39.00
SS-30 Wire Wrap Board B/C	$ 20.00
6802 Super CPU A/T	$235.00
6802 Super CPU B/C	$ 59.00
Video RAM, B/C $49.00, A/T	$195.00
Parallel I/O, B/C $49.00, A/T	$139.00
SS-50 Backplanes 4,6,8,12,&16 position @ $5.00 per slot w/o connectors	
SS-30 8 pos. BP w/o connect.	$ 39.00
Transition Card $49.00, A/T	$ 95.00
Molex Gold Male $1.60, Female	$ 1.60
Molex Tin Male $ .40, Female	$ .50

THOMAS INSTRUMENTATION

168 EIGHTH STREET — AVALON, N.J. 08202 (609) 967-4280

NJ RES. INCLUDE 5% SALES TAX

Allow 3 Weeks for Checks to Clear

CONT. USA INCLUDE $3.00 SHIPPING, CANADA $6.00, FOREIGN $12.00

MASTERCARD, VISA, and C.O.D. ACCEPTED

Send $20.00 to receive full documentation, schematics, & source listings for all boards currently in production.

'68' MICRO JOURNAL ADVERTISERS INDEX

This index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

THE CHIEFTAIN™ 5¼-INCH WINCHESTER HARD DISK COMPUTER

SO ADVANCED IN SO MANY WAYS . . . AND SO COST-EFFECTIVE . . . IT OBSOLETES MOST OTHER SYSTEMS AVAILABLE TODAY AT ANY PRICE.

● HARD DISK SYSTEM CAPACITY

The Chieftain series includes 5¼- and 8-inch Winchesters that range from 4- to 60-megabyte capacity, and higher as technology advances. All hard disk Chieftains include 64-k memory with two serial ports and DOS69D disk operating system.

● LIGHTNING ACCESS TIME

Average access time for 5¼-inch Winchesters is 70-msec, comparable to far more costly hard disk systems. That means data transfer ***ten-times faster*** than floppy disk systems.

● 2-MHZ OPERATION

All Chieftains operate at 2-MHz, regardless of disk storage type or operating system used. Compare this to other hard disk systems, no matter ***how*** much they cost!

● DMA DATA TRANSFER

DMA data transfer to-and-from tape and disk is provided for optimum speed. A special design technique eliminates the necessity of halting the processor to wait for data which normally transfers at a slower speed, determined by the rotational velocity of the disk.

● RUNS UNDER DOS OR OS-9

No matter which Chieftain you select . . . 5¼- or 8-inch floppy, or 5¼- or 8-inch Winchester with tape or floppy back-up . . . they ***all*** run under DOS or OS-9 with ***no need*** to modify hardware or software.

● UNBOUNDED FLEXIBILITY

You'll probably never use it, but any Chieftain hard disk system can drive up to 20 other Winchesters and four tape drives, with a single DMA interface board!

● SMOKE SIGNAL'S HERITAGE OF EXCELLENCE

This new-generation computer is accompanied by the same ***Endurance-Certified*** quality Dealers and end-users all over the world have come to expect from Smoke Signal. And support, software selection and extremely competitive pricing are very much a part of that enviable reputation.

20-Megabyte Tape Streamer Back-Up Option

Available with all Chieftain hard disk configurations. This cartridge tape capability provides full 20-megabyte disk back-up in less than five minutes with just one command, or copy command for individual file transfers. Transfers data tape-to-disk or disk-to-tape. Floppy back-up is also available in a variety of configurations.

The Chieftain Computer Systems:

Here are the Chieftain 6809-based hard disk computers that are destined to change the data processing industry . . .

☐ **CHIEFTAIN 95W4**
4-megabyte, 5¼-inch Winchester with a 360-k floppy disk drive (pictured).

☐ **CHIEFTAIN 95XW4**
4-megabyte, 5¼-inch Winchester with a 750-k octo-density floppy disk drive.

☐ **CHIEFTAIN 98W15**
15-megabyte, 5¼-inch Winchester with a 1-megabyte 8-inch floppy disk drive.

☐ **CHIEFTAIN 9W15T20**
15-megabyte, 5¼-inch Winchester with a 20-megabyte tape streamer.

Write or call today for details (including the remarkably low prices) on the total Chieftain Series . . . and on dealership opportunities.

SMOKE SIGNAL BROADCASTING®
31336 VIA COLINAS
WESTLAKE VILLAGE, CA 91362
TEL (213) 889-9340

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

Telephone ( ) _____

Intelligent Serial I/O Processor Board Now Available

The GIMIX Intelligent Three-port RS-232C Serial I/O Interface can significantly increase throughput of a multi-user system by reducing the number of interrupts between user terminals and the host CPU. The Intelligent I/O Board accomplishes this by buffering data transfers between system and users and preprocessing of the data.

Appropriate on-board software and operating system drivers are required. Software and drivers for OS-9 Level 2 will be available shortly from GIMIX.

- INDEPENDENT ON-BOARD **2MHZ 68B09** CPU
- UP TO **20K** OF ON-BOARD MEMORY (EPROM and RAM)
- BUFFERED DATA TRANSFER BETWEEN HOST AND ON-BOARD CPU USING A **Z8038** FIO I/O INTERFACE UNIT
- THREE **RS-232C** SERIAL I/O PORTS **(6551As)** WITH SOFTWARE SELECTABLE BAUD RATES, WORD LENGTH, STOP BITS, PARITY

Standard Version Including 4K RAM (Without Software) $438.11

— ***PARALLEL VERSION COMING SOON*** —

Uniflex For GIMIX Winchester Systems

TSC will be providing UniFLEX compatible with GIMIX Winchester systems. The NEW versions of UniFLEX for use with the Winchester systems will be delivered on 5" media as well as 8" media.

GIMIX 30 Pin Prototyping Board Now Available

- Double sided with plated thru holes and gridded power and ground lines.
- 8 rows of pads on .100 x .300 centers: up to 41 fourteen pin ICs.
- Accepts standard 6, 8, 14, 16, 20, 24, and 40 pin DIP devices.
- The entire top edge has pads for .100 x .100 header (ribbon) connectors.
- Pads for solder connections or .100 center headers on all 30 bus lines.
- Accepts 3 TO-220 regulators, 1 on the +8V & 1 ea. on the +/− 16V lines.
- Provisions for decoupling caps distributed throughout the array.
- Can be used with wire wrap, wiring pencil, solder wiring, etc.

With gold bus connectors and heat sinks (unassembled) $38.33

Now Available From GIMIX

(U.S. & Canada Only)

THE WINDRUSH EPROM PROGRAMMER

- ★ Probably the most versatile EPROM PROGRAMMER available. Interface & software for EXORcisor - II (fully addressable) and S-50 bus systems.
- ★ PROGRAMS AND VERIFIES 2508/2708, 2516/2716 (SINGLE AND TRI-VOLT TYPES) 2532, 2732, 2732A, 2564, 2764 and the 128K TMS2528 (16K x 8) . . . . . WITHOUT ADDITIONAL 'PERSONALITY' MODULES . . . . . .
- ★ PROGRAMMER extends out to your work area via 5' of twisted pair cable.
- ★ EXTENSIVE COMMANDS MENU...MOVE DATE, READ, PROGRAM, VERIFY EPROMS, EXAMINE/CHANGE BUFFER, FORMATTED DUMP OF BUFFER, FILL BUFFER.
- ★ Fully documented user's manual w/schematics & theory of operation. Professionally finished PCBs w/solder resist & component overlay.
- ★ SOFTWARE AVAILABLE FOR FLEX 2/9, SSB, OS-9 (LVL 1 NOW, LVL 2 LATER) and MDOS...All source files supplied. Specify disk size please!

NOTE: One version is supplied FREE. Extra versions: $25.00 each.

S-30 Interface/Programmer/Baseplate/Cable $375.00
EXORcisor Interface/Programmer/Baseplate/Cable $395.00

1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510
TWX 910-221-4055

GIMIX INC.

FLEX & RS COLOR COMPUTER

If you are tired of playing games on your TRS-80C™ Color Computer, or find that you are handicapped by the limitations of the RS BASIC in trying to write a Program that will allow you to actually USE the Color Computer as a COMPUTER, YOU ARE READY TO MOVE UP TO THE **FLEX9™** Operating System. If you want to have REAL PROGRAMMING POWER, using an Extremely Powerful Business BASIC, PASCALs, C Compilers, a full-blown Macro Assembler with a Library capability so you are not continuously "reinventing the wheel", YOU ARE READY TO MOVE UP TO THE **FLEX9™** Operating System. If you would like to see if YOU REALLY COULD USE A COMPUTER IN YOUR BUSINESS, or begin to make your Computer start PAYING IT'S OWN WAY by doing some Computer Work for the millions of small businesses around you, such as Wordprocessing, Payroll, Accounting, Inventory, etc., then YOU ARE READY TO MOVE UP TO THE **FLEX9™** Operating System. How?? **DATA-COMP** has the way!

DATA-COMP's FLEX9™ Conversion for the TRS-80C™ ColorComputer was designed for the SERIOUS COMPUTER USER; with features like greatly increased Display Screens, WITH Lower Case Letters, so you can put a FULL Menu on ONE Screen, or see SEVERAL Paragraphs at the same time; with features like providing a FULL Keyboard so you have FULL Control of your Computer AND it's Programs NATURALLY, without needing a chart to see what Key Combination will give you what function; with USER ORIENTED functions to make using the Operating System natural, like having the Computer AUTOMATICALLY determine what type of Disk is being used in what type of Disk Drive and working accordingly, rather that you have to specify each and every thing for it, or like having the Computer work with the Printer you have been using all along without you having to tell the new Operating System what is there; etc., etc., etc.

DATA-COMP has everything you need to make your TRS-80C™ Color Computer WORK for YOU; from Parts and Piec s to Full, Ready To Use SYSTEMS. DATA-COMP designs, sells, services, and SUPPORTS Computer SYSTEMS, not just Software. CALL DATA-COMP TODAY to make your Computer WORK FOR YOU!

System Requirements

FLEX9 Special General Version w/Editor & Assembler (which normally sell for $50.00 ea.)	$150.00
F-MATE(RS) FLEX9 Conversion Rout. for the RS Disk Controller when purchased with Special General FLEX9 Sys.	$69.95
when purchased without the General FLEX9 Sys.	$79.95
Set of Eight 64K RAM Chips w/Mod. Instructions	$99.95
Color Computer with 64K RAM and EXT. BASIC	$549.95
Color Computer with 16K RAM	$375.95
Color Computer with 16K RAM and EXT. BASIC	$465.95

Now Available
Enhanced F-MATE Version 2.1

SPECIAL SYSTEM PACKAGES

64K Radio Shack COLOR COMPUTER, Radio Shack COLOR DISK CONTROLLER, a Disk Drive System, Special General Version of FLEX9™, F-MATE(RS)™ and a Box of 10 Doubl Density Diskettes; a COMPLETE, ready to run SYSTEM on your Color TV Set. $1249.95

DISK DRIVE PACKAGES, etc.

These Packages Include the Radio Shack Disk Controller, Disk Drives with Power Supply and Cabinet, and Disk Drive Cable:

PAK #1 — 1 Single Sided, Double Density Sys.	$499.95
PAK #2 — 2 Single Sided, Double Density Sys.	$769.95
PAK #3 — 1 Double Sided, Double Density Sys.	$599.95
PAK #4 — 2 Double Sided, Double Density Sys.	$949.95

PARTS AND PIECES

Radio Shack Disk Controller	$179.95
1 ea. Single Sided, Double Density Disk Drive	$249.95
1 ea. Double Sided, Double Density Disk Drive	$349.95
Single Drive Cabinet with Power Supply	$89.95
Double Drive Cabinet with Power Supply	$109.95
Single Drive Disk Cable for RS Controller	$24.95
Double Drive Disk Cable for RS Controller	$34.95
Micro Tech. Prods., Inc. **LOWER CASE ROM** Adapter	$74.95
Radio Shack BASIC Version 1.1 ROM	$34.95

SOFTWARE

CHESS 6809

Requires FLEX™ and one of the following CRT terminals

Now Runs On Any Type Terminal

Features:

- **Two display boards.**
- **Four levels of play.**
- **Point scoring system.**
- **Play white or black.**
- **Change or set-up boards/place positions.**
- **Forfeit move.**
- **Swap sides.**
- **Make move and swap sides.**
- **Change skill level.**
- **Stop and restart game.**
- **Solve 'Mate in 1-2-3-4' moves.**

$79.95 Specify 5" or 8" disk

This is one of the strongest CHESS programs running on any microcomputer, estimated USCF Rating 1600 + .

Note: Personal checks allow 3-4 week delivery.

DIET-TRAC Forecaster

A Diet Planning and Analysis Program

DIET-TRAC Forecaster is a program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C-P-F %) or grams of Carbohydrate, Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual.

Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individuals are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. When a weight goal is given (either gain or loss), and a calorie plan is agreed upon between the computer and the individual, the number of days to reach the weight goal is projected. The starting and ending rate of weight loss is calculated, and a daily calendar with each day's predicted weight for a 30-day period is printed.

FLEX VERSION	**$59.95**
UniFLEX VERSION	**$89.95**

PRINTERS

The Epson MX-80
$495.00
The Epson MX-100
$725.00

MX-70 $355.00 MX-80 FT $575.00

MEMORY

SWTPC-Motorola, MP32
32K Dynamic Memory Board
Assembled & Tested
1 MHZ - No extended addressing
Can be set up for $0-7FFF or 8000-FFFF

$149.95

MasterCard VISA

DATA-COMP SOUTH EAST MEDIA

P.O. Box 794 *HIXSON, TN 37343*

1-615-842-4601

Verbatim Diskettes.

5" Soft Sector Disks

Single Side Single Density	$2.75 ea
Single Side Double Density	$2.75 ea
Double Side Double Density	$4.92 ea
Plastic Storage Box	$7.00 ea

8" Soft Sector Disks

Single Side Single Density	$3.75 ea.
Single Side Double Density	$4.10 ea.
Double Side Double Density	$4.75 ea.
Plastic Library Box	$5.00 ea.

Foreign Orders Add 10% Surface—20% Air Mail

DRIVES & CABINETS W/PS

5 1/4"

TANDON - Single Sided, Double Density, 40 Track.....$249.95

TANDON - Double Sided, Double Density, 40 Track.....$349.95

CABINET - Single Drive with Power Supply.....$ 79.95

CABINET - Double Drive with Power Supply.....$ 99.95

CABLE - Single Drive.....$ 24.95

CABLE - Double Drive.....$ 34.95

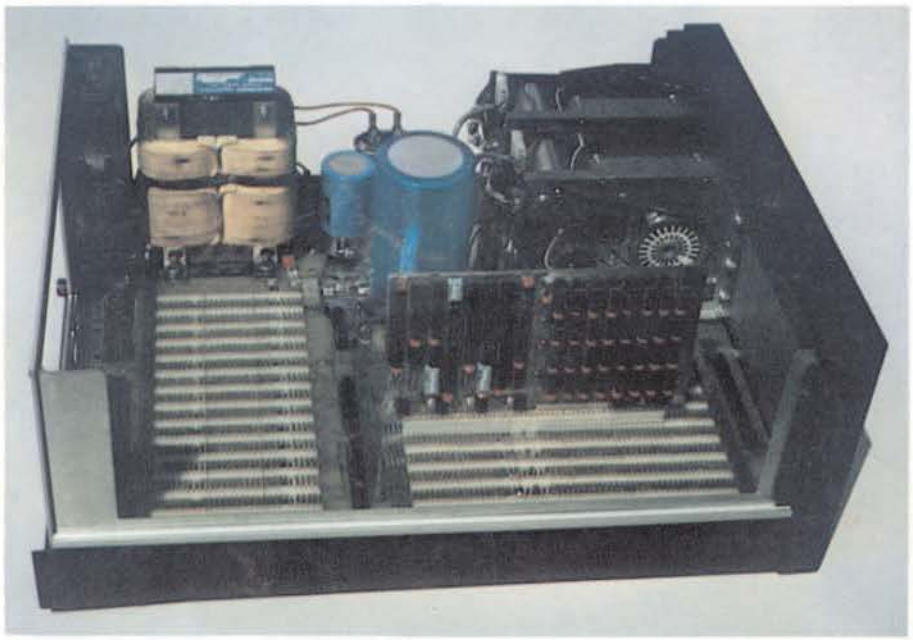
NOTE - When ordering cables please specify S50 Bus or Other!!!

Call or write for disk controller Board information.

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025

HELIX™



THE MAINFRAME

- Industry Standard Optima™ Cabinet
- Largest Constant Voltage Power Supply in the Industry
- S-64 Bus gives 16 Bit Power and S-50 Bus Compatibility
- 10 Main (S-64) Slots
- 14 I/O (S-30) Slots plus 2 On-board
- On-board Baud Rate Generator to 38.4Kb
- Space and Power for two 5¼" Disk Drives
- Full Address Decoding for I/O Slots
- Two RS-232 Serial and Two parallel Ports On-board
- Single Board Construction for Reliability
- Faraday Shielded Bus Lines give "Text Book Clean" Signals

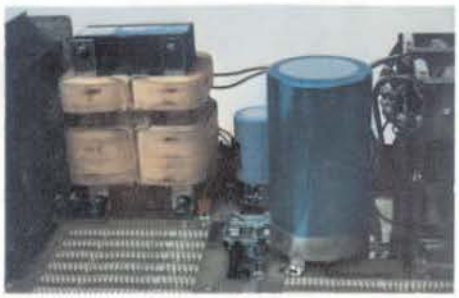
THE PROCESSORS

6809

- Standard 2 MHz Operation
- Standard DAT Compatible with GIMIX and SWTPC
- Standard 6840 Interval Timer
- Standard 1K Scratchpad RAM
- Standard Clock/Calendar with Battery
- Provision for Programmers Console

68000

- Standard 8 MHz Operation
- Memory Management Hardware
- Provision for Programmers Console
- 16 Bit Power and 8 Bit Compatibility



The HELIX™ computer system represents the latest advance in S-50 bus computer systems. Relying on the physical nature of S-50 bus connectors to guarantee compatibility, the HELIX adds 14 bus lines (becoming S-64) to allow a 68000 processor to operate with full 16 bit data transfer and 24 bit addressing, while at the same time providing full interchangability with existing S-50 components.

Offered with a selection of processors, memories, and peripheral controllers, a HELIX system can be configured for applications ranging from advanced hobbyist to multiterminal time-sharing.

Designed to offer the utmost in speed, reliability, and utility at a reasonable price, it represents a new standard of quality for those who require a professionally designed computer for professional use.

THE POWER SUPPLY

- Ferro-resonant Transformer for Line Noise and Under-Voltage Protection
- Conservative 25 Amps at 8.5 Volts
- Conservative 5 Amps at ±16 Volts
- Conservative Component Rating for Reliability

THE COMPONENTS

- Fully Socketed
- Gold Plated Bus Connectors
- Only "8" Series 68XX Components Used
- Only Top Grade Logic Circuits Used
- Industrial Grade Components Throughout

THE MEMORIES

DM-64

- Field Proven
- Proprietary Memory Control Logic
- Fully Transparent Refresh
- Tested at 2.5 MHz Operation

DM-512

- 512K Bytes on a Single S-64 Board
- 16 Bit Power and 8 Bit Compatibility
- Runs in Existing S-50 Systems where Physical Space Allows
- Full 24 Bit Addressing
- Fully Transparent Refresh

THE PRICES

Because of the variety of configurations possible, full pricing cannot be given. Representative prices are:

- 64K 6809 HELIX $1995
- 64K 68000 HELIX $2595
- 512K 6809 HELIX $4450
- 512K 68000 HELIX $4995

HAZELWOOD COMPUTER SYSTEMS

907 E. Terra, O'Fallon, Missouri 63366 (314) 281-1055

Dealer and OEM Inquiries Invited. We support our Dealers.

Optima is a Trademark of Scientific-Atlanta